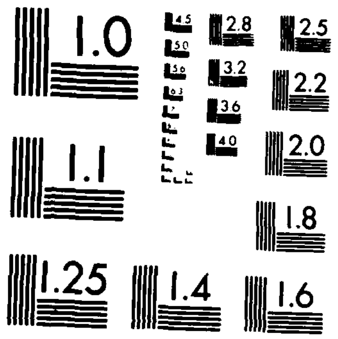
AD-A141 049   PASCAL STATISTICAL PROCEDURES PACKAGE (PSPP)(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGINEERING D P KUNKEL DEC 83 AFIT/GSO/OS/83D-4   1/3

UNCLASSIFIED   F/G 9/2   NL



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

AFIT/GSO/OS/83D-4

PASCAL STATISTICAL PROCEDURES PACKAGE (PSPP)

THESIS

David P. Kunkel
Captain, USAF

AFIT/GSO/OS/83D-4

AFIT/GSO/OS/83D-4

PASCAL STATISTICAL PROCEDURES PACKAGE (PSPP)

THESIS

Presented to the Faculty of the School of Engineering

of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the

Requirements for the Degree of

Master of Science in Space Operations

David P. Kunkel, MBA

Captain, USAF

December 1983

Preface

The purpose of this study was to develop a set of procedures that could be combined into a multivariate data analysis package that would run on an Apple microcomputer. The immediate use for this package is as a teaching aid in the classroom or microcomputer center and as a research tool for users to do a 'ball-park' analysis of a data base.

Included in the package are procedures to handle data base definition and modification, Factor (Principal Component) analysis, and Canonical Correlation analysis. Because the package was constructed in discrete units, the data section and applicable parts of the statistical sections can be incorporated into other multivariate routines.

In writing this package I have had a lot of help from others. I am thankful to my advisor, Major Joe Coleman, for his help in interpreting the statistical formulations used. I also wish to thank my reader, Captain Patricia Lawliss, for her many hours in helping to debug the programs as well as helping construct the package structure.

David P. Kunkel

Table of Contents

List of Figures

List of Tables

Abstract

This study showed that a set of procedures could be written and combined into a multivariate data analysis package that will run on a microcomputer. This package can be used as a teaching aid in the classroom or microcomputer center and as a research tool for users to do a 'ball-park' analysis of a data base. Included in the package are procedures to handle data base definition and modification, Factor analysis, and Canonical Correlation analysis.

The PASCAL Statistical Procedures Package (PSPP) was written on an Apple IIe microcomputer using the Apple PASCAL language and operating system. It will output to a printer in a 132 character per line format. If an on-line printer is only capable of 80 characters per line, wrap-around will occur.

The package is composed of 4 top-level procedures stored in Regular Units and 163 library procedures stored in 13 Intrinsic Units. Units are Apple PASCAL structures that allow for program segmentation.

I. Introduction

Statistical analysis of data, in order to suggest possible cause and effect processes, has long been an accepted methodology. Only recently have multivariate techniques become accepted as a means of reducing error introduced by the interdependence between the presumed independent variables. One of the most common software packages for working with multivariate data is SPSS (Statistical Package for the Social Sciences) by Nie (17) that has been available for some time for use on mainframe computers. However, there is a lack of software for microcomputers which are portable enough to be used in the classroom.

Statement of Issue

SPSS is written for mainframe computers and can utilize large amounts of core memory. It is designed to produce a great number of combinations of statistics for large data bases. It is not user friendly and has slow turn around. Consequently, it is not a good tool to teach multivariate techniques to the new student.

Research Question

This thesis is an attempt to write a package of multivariate routines, to be used in an interactive user friendly program, that can be run on a microcomputer in the classroom or microcomputer center. The question to be answered is whether a useful set of routines can be written that can run quickly and accurately on a microcomputer.

<u>Objectives of the Research</u>

1) Write routines to do data input and modification, Canonical Correlation analysis, and Factor analysis, in PASCAL for the microcomputer.

2) Write a User's Guide for the package.

3) Validate the procedures by comparing the results to those achieved via SPSS.

<u>Specific Objectives</u>

1) The routines should be user friendly to help the unsophisticated user.

2) Swapping in and out of core to disk is required to minimize core usage by the routines in order to maximize the amount of data that can be analyzed.

3) Sophisticated matrix manipulation techniques need to be used for speed and to minimize core usage.

4) Numerical analysis techniques need to be used to approximate higher order polynomials to at least the tenth order for necessary flexibility.

<u>Scope</u>

The thesis consists of four major sections: the main body and three appendices. The main body introduces the problem and discuss the procedures used for and the results of the program validations. The first appendix contains the User's Guide, the second is the results of the validation runs, and the third is the coding for the package.

The User's Guide consists of an introduction and six sections. The introduction outlines the package as to the techniques available and the kind of data that can be analyzed. The first three sections specifically outline usage of each of the three main modules: DATA, CANCOR, and FACTOR. The fourth covers formatting of blank disks for storage of data files. The next outlines special features to manipulate data files, and the last describes the specific construction of the package.

This package is limited as far as the size of data b [illegible]es it will handle (200 records of 10 variables each). Accu[illegible] is limited to single precision (6 or 7 significant digits) and while the routines are precompiled, the program is slow in execution compared to a mainframe computer. These limitations are driven by the nature of microcomputers and their operating languages. However, because the package is interactive, results are available to the user immediately in a user friendly manner.

II. Background Study

Introduction and Organization

In order to meet the objectives and subobjectives, it was necessary to research five topic areas. First, to present the proper use of each of the techniques, as well as insuring correct procedures, multivariate data analysis in general was researched. Next, because all of the techniques are based on matrix algebra, matrix manipulation techniques were researched for use in the various procedures. Third, in order to use matrix algebra, it is necessary to solve an eigenvalue problem. Because the computer can not solve it analytically, polynomial approximations must be used. To be as flexable as possible, it should be capable of at least tenth order polynomial approximations. Numerical analysis techniques were researched for the necessary background to solve such a problem. Next, due to the small core memory limitation inherent in microcomputers, it was necessary to research both program and data segmentation procedures. Finally, because the primary use of this package will be for classroom instruction, it is necessary that the routines be user friendly and operational in an interactive mode. Computer menus are a key factor in program useability as is 'idiot-proofing' routines so that the user is prevented from making critical errors.

Multivariate Analysis

The bulk of the material used as the basis for procedure construction comes from information presented in the AFIT course 'Applied Multivariate Data Analysis.' McNichols (13) presents the mathematical background and a step-by-step development for each of the techniques. Class notes (4) supply supplemental and clarifying information. The SPSS manual (17) contains a limited background for the techniques, in addition to procedures for running them on the CDC 6600 mainframe computer for validation.

Matrix Manipulation

McNichols (13) presents some of the classical matrix algebra, but little on actual implementation. Specific procedures for multiplication and inversion were found in Carnahan (3), Conte (5), and McMillan (12). The numerical analysis texts (3,5) also contain necessary checks to insure invertability.

Numerical Analysis

McNichols (13) has a good presentation of the eigenvalue problem, but does not get into polynomial approximating techniques. Carnahan (3), Conte (5), and Douglass (7) provide the necessary background and procedures to solve a tenth order approximation.

Memory Maintenance

Because the compiled program code and data storage locations together would use more memory space than is in a microcomputer, it is necessary to split either the data or the program into segments. Data structures needed to be developed to minimise the use of core by overwriting the same memory location whenever possit'e. It was lso necessary to not duplicate variable structures by usin 'call by location' procedures as opposed to 'call by value' outines. Lewis (10) contains several procedures for develop.ng such structures. The swapping of data between core and disk requires specialized interface routines which can be found in Swanson (19).

The PASCAL programming language allows the usage of routines that can split a program into 'units' that are stored in 'libraries' and are present in memory only when needed. Merritt (14,15,16) provides an excellent explanation and guide for usage of those routines. General information about PASCAL is found in the Apple PASCAL manuals (1,2) and Zaks' Introduction to PASCAL (20).

Interactive Driver and Graphics

As mentioned above, ease in program useability is centered around effective computer menus. Root (18) presents to the public domain a powerful procedure for developing customized menus. Some aspects of his procedure were useful for data input as well as aiding in user selection of program options.

In order for programs to continue executing despite any errors made by the user, it is necessary to 'idiot proof' the software. Cox (6) presents several techniques that protect both the program and the disk.

Finally, user friendly programs need to present solutions in a format that is easily understood and interpreted by the user. Graphics displays do this much better than lists of numbers. Procedures for numerous graphics generations are found in Korites (9).

Summary

A review of statistical and microcomputer journals failed to turn up programs of a multivariate nature that could do Canonical Correlation or Factor analysis on a microcomputer. The materials indicated above were sufficient to solve the research problem in question: the development of a multivariate analysis package for use in the classroom on a microcomputer. With this program an instructor can teach the techniques necessary for the student to run multivariate data analysis programs, such as SPSS, on main-frame computers with larger data bases. This will not only increase student understanding of multivariate analysis but will also decrease the time spent learning the techniques on non user friendly systems.

III. Package Validation

In order to check the validity of the numbers produced by the package, a cursory comparison was made between the output of the package and the output of SPSS while using the same data bases. Included in Appendixes A & B are the data and outputs from both the package and SPSS.

<u>Validation Runs</u>

The first section of Appendix B is the SPSS output for the data used as calculation examples throughout the User's Guide (Appendix A). Both the results of CANCOR and FACTOR are represented.

The next section includes data bases from an original Computer Performance Analysis data base that was 164 records long by 10 variables wide. The data base was first edited by deleting one record that was not representative of the rest. Next, four original variables plus two computed variables (numeric sums of two original variables) were used to make a CANCOR data base. Finally, six original variables were used to make a FACTOR data base.

Both of the large data bases were then run through both the package and SPSS and the results placed in the Appendix B. It should be noted that SPSS, which keeps track of more significant digits, prints out more decimal places than PSPP. For comparison, round the SPSS outputted values to the same number of decimal places printed by PSPP.

Example CANCOR Data

A comparison of the SPSS output with the calculations produced by the PSPP showed identical values for all areas common to the two packages with two exceptions. First, the CHI Square test statistics differ slightly. This discrepancy is probably the result of different calculation routines coupled with the differences in accuracy due to significant digit storage. The PSPP results are comparable in magnitude but more conservative (smaller) than those produced by SPSS.

Second, the Coefficients for Canonical Variables output of both sets show a sign reversal for CANVAR 2. This would be the consequence of a sign reversal of all the values in the eigenvector associated with the second eigenvalue. This is probably the result of different calculation routines coupled with the differences in accuracy due to significant digit storage. It should be noted that this sign reversal also shows up in the Canonical Variate Scores and the Structure Correlations outputs but has no impact on the Indexes of Redundancy because the associated Alpha and Beta values are squared.

Example FACTOR Data

A comparison of the SPSS output with the calculations produced by PSPP showed identical values for all areas common to the two packages.

CANCOR Validation Data

A comparison of the SPSS output with the calculations produced by PSPP showed identical values for all areas common to the two packages with three exceptions. As before, the CHI Square test statistics differ slightly (first or second decimal place) with the PSPP results being more conservative (smaller) than those produced by SPSS. Next, the Coefficients for Canonical Variables output of both sets show a sign reversal for CANVAR 1. As before, this would be the consequence of a sign reversal of all values in the eigenvector associated with the first eigenvalue. Lastly, two of the Coefficients for Canonical Variables in CANVAR 3 of the Second Set differ slightly in the fourth decimal place. This is probably due to the differences in accuracy due to significant digit storage.

FACTOR Validation Data

A comparison of the SPSS output with the calculations produced by PSPP showed identical values through the eigenvalue outputs where there is a difference in the fourth decimal place for two of the six eigenvalues. Further calculations based upon these eigenvalues show an increasing divergence between the SPSS and PSPP outputs. The Factor Matrix output has a worst case divergence in the fourth decimal place, while both the Communality and Factor Score Coefficients outputs have worst case divergences in the third decimal place. No comparison was done of the Factor Scores, but they would probably be comparable through at least the second decimal place.

There is also a sign a reversal in the second and third Factors in both the Factor Matrix and the Factor Score

Coefficients outputs. As before, this would be the consequence of a sign reversal of all the values in the eigenvectors associated with the second and third eigenvalues.

Conclusion

The results of the calculations done by PSPP are very close to those done by SPSS. The discrepancies are probably the result of different calculation routines coupled with the differences in accuracy due to significant digit storage. Because the data in the bigger data bases start with a large number of decimal places, subsequent calculations are constantly being truncated when more than 6 or 7 significant digits are produced.

It should be noted that the PSPP runs were made using the most accurate setting of Epsilon when the eigenvalues were calculated. This setting requires more iterations, and consequently more time; especially for larger matrices. If a larger size for Epsilon is used, the eigenvalues and all subsequent calculations based on those eigenvalues differ from those produced by SPSS. It is up to the user to decide which is more important: the speed in calculating the eigenvalues or their accuracy. The default Epsilon of .0001 was chosen as a good trade off value between speed and accuracy. The eigenvalues calculated using the default appear to be accurate through at least two decimal places.

IV. Recommendations

The future will most likely see an ever increasing use of computers in educational settings as an aid to learning in all disciplines. Consequently, the need for software to run on portable microcomputers will also increase. This study has shown that it is possible to write a package of routines that can be used by an instructor in the classroom or microcomputer center in teaching multivariate data analysis.

Further research is recommended in this area using the procedures in this thesis as a starting point. It should be possible to write more multivariate routines that use the data section and applicable parts of the other sections. Routines to do Multiple Analysis of Variance (MANOVA), Multiple Regression, Residual Analysis, and Discriminant Analysis are a few areas that could be added.

Improvements could also be made within the body of the current code. First, the use of PASCAL Long Integer variables instead of Real variables could greatly increase the accuracy and flexability of the package at the expense of speed, memory, and disk space. Real number storage, while simpler and cheaper to use, is limited to 6 or 7 significant digits on the Apple in PASCAL. Use of Long Integers would allow string storage and manipulation of alphanumeric characters with up to 36 significant digits of accuracy with numbers. Also, it should be possible to add more and better high resolution graphics to the package at the expense of memory and disk space.

These last areas of improvement may be infeasible because the package is already pushing the limits of useable memory (39,900 bytes) available in a standard Apple microcomputer capable of running PASCAL.

Appendix A

PASCAL

STATISTICAL PROCEDURES

PACKAGE

(PSPP)

USER'S GUIDE

by

David P. Kunkel

Table of Contents

Introduction

Multivariate Statistics

Statistical analysis of data, in order to suggest possible cause and effect processes, has long been an accepted methodology. Only recently have multivariate techniques become accepted as a means of reducing error introduced by the interdependence between the presumed independent variables. The PASCAL Statistical Procedures Package (PSPP) is a collection of routines that can handle data input, storage, and manipulation plus the two multivariate techniques of Canonical Correlation analysis and Factor (Principal Component) analysis. Some user knowledge is expected about the operation and booting of microcomputers.

Data Structure

The PSPP is designed to work only with real data values. All integer inputs are stored as reals. Alphabetic entries must be transposed to numerals; A/B/C could be entered as 1/2/3 or -1/0/1, as desired. Further, variables should be pre-scaled so that they are not very large or very small. There is a limit of 7 significant digits in internal storage. Larger numbers will be rounded and represented in scientific notation. Entries should be rescaled to representations of millions of units or hundredths of a unit, as applicable. It is important to note that arithmetic operations, such as the calculation of means, done on data transposed from the nominal (e.g. male / female) type or the ordinal (e.g. high-school / under-graduate / graduate) type could produce meaningless numbers. Finally, there are no provisions to handle missing data. A number must always be entered; even if only a zero. If this would bias results, consideration should be given to excluding cases with missing data fields.

Limitations

The PSPP is limited in the size of data bases it can handle. There is an upper limit of 200 records or cases of data, each with a limit of 10 variables or data fields. Analysis of larger data bases should be done via SPSS on a mainframe computer. There is a limit of 80 columns or characters per record so that one entire record can be viewed on one line of the microcomputer screen. There is a minimum of 8 columns and a maximum of 15 columns for each field width (variable). This includes room for 6 significant digits, a leading minus sign, a decimal point, scientific notation if required, and leading spaces. Data should be pre-scaled as described above to meet this requirement. Finally, there is an upper limit of 15 characters in the storage of field names. Longer names are automatically truncated on entry. Further, the names are truncated down to their designated field width when displayed on the screen. This means that the name 'EDUCATION' will be displayed as 'EDUCATIO' if the field width is 8. It should be noted that if a printer is being used, data lines are expanded up to 132 columns with spaces inserted between the fields and longer names printed. If the printer is limited to 80 columns, wrap-around will occur if more than 80 characters are printed on a line.

<u>System Start-up</u>

The PSPP is written to be run on an Apple IIe system with two disk drives and printer connected. It will run without a printer. The program disk should be inserted in the boot drive, Side 1 up. A preformatted data disk should be inserted in the non-boot drive. After booting, the sequence in Figure 1 should be followed to run PSPP.

```
>COMMAND: E(DIT, R(UN, F(ILE, C(OMP, L(INK, X(ECUTE,..
 S

>SWAPPING IS OFF
>TOGGLE SWAPPING
 Y

>COMMAND: E(DIT, R(UN, F(ILE, C(OMP, L(INK, X(ECUTE,..
 X

>EXECUTE WHAT FILE?
 PSPP  (return)
```

Figure 1. System Start-up

Use of SWAPPING allows for more memory use. It is required if the MODIFILE procedure in the DATA module is going to be used.

In Figure 1, and throughout the rest of the User's Guide, computer prompts and messages are proceeded with a '>' character. All others are as the user would enter at the keyboard; with notes in parenthesis.

Introductory remarks, if selected after booting, explain how to make entries. There are two types of user inputs:

1 - When asked to 'ENTER' a value, the user should type a response, the press the (return) key;

2 - When asked to 'SELECT' an option (from a menu or list of options) or asked a YES or NO question, the (return) key need not be pressed.

Whenever a routine finishes, control is returned to a higher level menu and the user must then select how to proceed.

In order to format blank data disks, reference Section IV. Other special procedures for manipulating the data disks are explained in Section V. Included are procedures that L (list directory), R (remove a file), C (change a file name), and K (crunch available space).

I. Data Manipulation

The Data Manipulation Module (DATA) is the largest and most complicated part of the PSPP. It handles the input of new data into a data file, saving that data file to disk, loading that data file from disk, the modification of data in a data file, and the printing or echoing of the data file to either the printer and the screen, or just the screen. Before proceeding, the user should reference Section IV on the Formatting of Blank Data Disks so that any new data files made can be saved.

Make File Routine

Once selected, MAKEFILE gives the user the option of viewing the following instructions or proceeding immediately to the GATHERDATA routine. Before entering data, the user must modify the data with certain considerations in mind. First, all entries must be numeric because data is stored in a real array. Nominal categories such as Male/Female that have been coded as M/F, or even A/B, need to be recoded as numbers, such as 1/2. Missing or blank fields in a record might be stored as zero, but consideration should be given to excluding that record if that would bias desired results. Next, there are upper limits of 200 records per data file and 10 fields per record. There is a further limit of 80 characters per record. This includes all decimal points and spaces between fields. This is done to allow the viewing of one complete record on an 80 column screen. Lastly, the first field of any record cannot be 9999, as this is the value used to signify data entry completion.

The data is entered into the file in three stages. First, the number of data fields or variables is requested. This value can be any number between 1 and 10 inclusive, but only the integer part is saved. The user is then shown the value the computer accepted and is given the option of changing it. For instance, if the user inputs 6.7, the computer will accept 6 as the number of fields.

The name and width of each field is then requested. The user must remember to leave room for the largest (in number of characters) value in each field, as well as decimal points and spaces between fields. There is a minimum limit of 8 and a maximum of 15 characters per field. To calculate field width, take the number of characters desired left of the decimal point and add 7. For example, if 23.45 is the largest value in a field, set the field width to 9 (i.e. 2+7).

The field name should be less than or equal to the field width; otherwise the name will automatically be truncated to fit. For example, 'EDUCATION' will be stored as 'EDUCATIO' if the field width is 8. Once all names and widths have been entered, the user is given the option of making changes until the desired structure is achieved.

Finally, each record is entered, one field at a time. After the last field is entered, the user is asked if any changes need to be made before moving on to the next record.

It should be noted that the computer does not check whether the field width was violated by any entry. This will not affect any computations, but when the data is echoed, the data will not be printed in neat columns and wrap-around on the screen may occur if there are more than 80 columns of data. After the last record has been entered, and the number of records (NUMREC) is less than 200 (MAXREC), the value 9999 should be entered into the first field to signify data entry completion. After the last record has been entered and approved by the user, control is returned to the main Data Module menu.

Figure 2 is an example of how these criteria can be met, given the case where sex and letter grades of five students need to be entered.

Name	Sex	Grade		StuNum	Sex	Grade
Mike	M	B+		1.00000	1.00000	3.30000
Sally	F	B		2.00000	2.00000	3.00000
Dave	M	A-	==>	3.00000	1.00000	3.70000
Donna	F	A		4.00000	2.00000	4.00000
John	M	C		5.00000	1.00000	2.00000

Figure 2. Sample Input

Save File Routine

Before this routine is run, the user should ensure that a properly formatted data disk is in the non-boot disk drive, and that there is enough room on that disk. There is room on one disk for about 14 different data files, if all are the maximum size of 200 records by 10 variables -- more if the files are smaller. Files are automatically written at the beginning of the largest free area on the disk. It should be noted that changing a file and then resaving it with the same name will cause the relative positions of files on a disk to change. This could lead to a subsequent save failure due to lack of space, if the largest available area on the disk is smaller than the file size. See Section V for instructions on 'Krunching' data files to consolidate available space.

When activated by the user, SAVEFILE asks the user to enter the desired file name. The computer will treat lower case and upper case letters the same. There is a limit of 10 characters in the file name and the first character must be a letter. The data file is then saved to a properly formatted data disk under the name: <user inputed name>.TEXT. The .TEXT suffix is used only by the computer and should NOT be used by the user. When the save is successfully completed, or the user declines to try another save after a failure, control is returned to the main Data Module menu.

<u>Load Data Routine</u>

When activated by the user, LOADDATA asks the user to enter the desired file name. If the desired filename or disk is not found, the user is notified of the failure and offered a chance to try again. Once the specified file is found, the load begins and overwrites any data previously existing in the data arrays. When the load is successfully completed, control is returned to the main Data Module menu.

Modify Data Routine

The MODIFILE routine is the largest and most complex section of the Data Module. Once a data file has been loaded into memory, either by the user with MAKEFILE or from disk using LOADDATA, the various parts of MODIFILE can be used to add a record, delete a record, add a field, delete a field, change a record, or change a field. It should be noted that additions cannot be made that would violate the upper limits of 10 fields per record or 200 records per file. NOTE: The SWAPPING option should have been set when the system was booted in order for MODIFILE to run properly.

Add a Record. When activated by the user, ADDAREC has the user enter a record one field at a time in the same manner as used in the MAKEFILE routine. Once all fields have been entered, the user has the option of making changes until the record is acceptable. Once accepted, the record is stored at the end of the data array.

Delete a Record. When activated by the user, SUBAREC asks the user the index of the record to be deleted. This number must be between 1 and NUMREC (number of records in file). The selected record is then displayed for the user who has the option of either proceeding with or canceling the removal of that record. If the removal is accepted, that record is overwritten by the last record in the file and NUMREC is decreased by one.

Add a Field. When activated by the user, ADDAFLD has the user define the new field width and name in the same manner used by the MAKEFILE routine. A check is made to insure that the upper limit of 80 characters per record is not violated. Once properly defined, the new field is filled by the FILLFIELD routine described below.

<u>Delete a Field</u>. When activated by the user, SUBAFLD displays the fields and widths of those that are currently in the file. The user is then asked the index of the field to be deleted, if any. This number must be between 1 and WIDTH (number of fields in a record). If one is selected, that field is overwritten by the last field in the file and WIDTH is decreased by one. A warning is displayed if the last field in the file was deleted.

<u>Change a Record</u>. When activated by the user, CHGAREC asks the user the index of the record to be changed. This number must be between 1 and NUMREC (number of records in file). The selected record is then displayed for the user who has the option of either proceeding with or canceling the change of that record. If a change is designated, the user has the option of making changes until the record is acceptable.

<u>Change a Field</u>. When activated by the user, CHGAFLD displays the fields and widths of those that are currently in the file. If a change is still desired, the routine has the effect of deleting the selected field and then adding a field in that position in the same manner as ADDAFLD. Once properly defined, the new field is filled by the FILLFIELD routine described below.

<u>Fill Field</u>. When activated by either the ADDAFLD or CHGAFLD routines, FILLFIELD gives the user three options for filling the designated field and definitions of each. The Recode option fills the specified field with user-selected constants; based on partition(s) within that or a different field. The Compute option computes and stores in the specified field the results of one or more arithmetic operations on one or more fields. The User Select option accepts data as input by the user at the keyboard, one record at a time.

<u>Recode</u>. When activated by the user, RECODE first displays a set of instructions. The routine works by partitioning the data of a selected field based on range(s) between two endpoints. The user has the option of entering numeric endpoints or using the values LOWEST and HIGHEST. Those points indicate the two extremes of the data field. It should be noted that once started, the user cannot leave the RECODE routine without using LOWEST and HIGHEST at least once. This is done to ensure all data points in the field are recoded. The routine is repeated as many times as desired by the user, but no actual recoding is done until the user exits the routine. At that time, the NEWFIELD buffer, where all recodes are temporarily stored, is written over the specified field. Figure 3 shows how a recode session might look. When exiting RECODE FIELD, after having set LOWEST and HIGHEST at least once, the user has one last option of making the save final or exiting without saving.

```
>Enter field to use in recoding:
 4        (return)

>Select desired option:
        >1 - Enter a partition
        >2 - Exit RECODE FIELD
 1

>Set partition bottom edge using:
     >1 - Numeric endpoint
     >2 - LOWEST value
 2

>Set partition top edge using:
     >1 - Numeric endpoint
     >2 - HIGHEST value
 1

>Enter upper endpoint:
 12.0     (return)

>Enter value to recode partition with:
 1         (return)

>Partition is:
             >Recode LOWEST to 12.00000 with 1.00000

>Select desired option:
        >1 - Proceed with RECODE
        >2 - Skip this RECODE
 1

>Recoding. . .

>Select desired option:
        >1 - Enter a partition
        >2 - Exit RECODE FIELD
 2

>WARNING:  Must reference both HIGHEST and LOWEST
          >Press any key to continue      (return)

>Select desired option:
        >1 - Enter a partition
        >2 - Exit RECODE FIELD
 1
                        .
                        .
                        .
```

Figure 3. Sample Recode Session

<u>Compute</u>. When activated by the user, COMPUTE displays some instructions prior to proceeding. This routine works by performing a computation based on one or two fields and/or user inputted constants and one operand. Any undefined results will be stored as 99.9999. The procedure can be executed more than once for two or more operations with the specified field holding the intermediate value(s). The user is first asked whether to use a field or a number for the first variable. Depending on selection, the index of the field or the value of the number is then entered. Next, one of the operands from Figure 4 is selected by the user.

These require a second variable:	
A - Addition	(+)
B - Subtraction	(-)
C - Multiplication	(*)
D - Division	(/)
These operate on the first variable:	
E - Square	(SQR)
F - Square Root	(SQRT)
G - Natural Log	(LN)
H - Log Base 10	(LOG)
I - Exponential	(EXP)
J - Absolute Value	(ABS)
K - Truncate	(TRUNC)
L - Round	(ROUND)

Figure 4. COMPUTE Operands

If the operand requires a second variable, it is entered in the same manner as the first. The computation selected by the user is then displayed for final approval before any computation is made. As an example of how this procedure might be used, assume that it is desirable to multiply the contents of Field #3 by 2, add the contents of Field #7, and store the results in Field #4. During the first time in COMPUTE, Field #3 is designated for Variable #1, the multiply operand is selected, the value 2.0 is designated, and the resultant is stored in Field #4. During the second time in COMPUTE, Field #4 is designated for Variable #1, the addition operand is selected, Field #7 is designated for Variable #2, and the resultant is stored back in Field #4.

User Input. When activated by the user, USERINPUT displays a warning about inputting values that exceed the MAX WIDTH for the field. If there is a value too wide, the user should enter the rest of the values, run the CHGAFLD routine to expand the field width, and then exit FILLFIELD without changing the field values. If the user decides to continue with USERINPUT, the values are then input one record at a time. The current record index and the total number of records (NUMREC) are displayed. Any errors on entry should be noted and later corrected using the CHGAREC routine.

Once completed, all of the six parts of MODIFILE return control to the MODIFILE menu, where the user has the option of further modifications or exiting the routine. After exiting MODIFILE, control is returned to the main Data Module menu.

Echo File Routine

When activated by the user, ECHOFILE asks the user if a limited or complete echocheck is desired. If a limited one is selected, the user is given the option of which fields to print. After selection, the user is then given the option of making changes until satisfied. Once the fields have been finalized, the user is given the option of sending the echocheck only to the screen or to the screen and printer (if there is one available). The file is then echoed, one page at a time, with the user pressing any key to display the next page.

II. Canonical Correlation

The Canonical Correlation Module (CANCOR) has the user select two sets of variables from the data array. It then derives a linear combination from each set so that the correlation between the two linear combinations is maximized. In other words, the goal is to account for a maximum amount of the relationship between the two sets of variables. It is similar to a multiple regression problem with more than one criterion variable as well as more than one predictor variable. The data in Table I is the data used in the examples throughout this section.

Y1	Y2	X1	X2
1	3	2	4
3	2	4	3
4	6	5	7
5	3	6	4
7	5	8	6
6	8	7	9
9	7	6	8
8	9	9	7
5	7	3	6
9	4	9	6

Table I. Example CANCOR Data

Assign Variables. When first activated by the user, CANCOR calls ASSIGNVARS for variable selection. The user is then shown the current status for each of the fields and asked whether to assign a predictor, assign a criterion, remove an assignment, or exit ASSIGNVARS. Once entered, this routine will not allow the user to exit until at least two variables are assigned to each set. This means that there should be at least four variables in the data base before calling CANCOR. The user has the option of making as many changes as needed to correctly assign the variables to the two sets.

Calculate Statistics. The next routine called by CANCOR calculates and prints the means and standard deviations for each of the selected variables, segregated by set. If a printer is on-line, the results are printed there as well as on the screen. Figure 5 is the output using the example data.

Variable	Mean	Standard Deviation
Y1	6.00000	2.48633
Y2	5.75000	2.37888
X1	6.33333	2.42462
X2	6.41667	1.97523

Figure 5. Statistics Output Example

At this point, the user is given the option of exiting CANCOR without proceeding to data standardization.

Standardize. In preparation for calculating the Sample Correlation Matrix, the selected variables are standardized by subtracting the field mean and dividing by the field standard deviation. It should be noted that if there is only one record, the standard deviations are zero, the data values become undefined, and are represented as 99.9999. The standardized values are written over the original values in the data array, but are not automatically saved to disk.

<u>Generate Correlation Matrix</u>. The next routine calleo by CANCOR generates and prints the Sample Correlation Matrix of the designated fields by first generating smaller first and second set self-correlation matrices and the first/second set cross-correlation matrix. These partitions are then stored in CORRMATRIX as shown in Figure 6. It should be noted that the

```
              ¦  R(YY)   ¦  R(YX)   ¦
CORRMATRIX =  ¦----------¦----------¦
              ¦  R(XY)   ¦  R(XX)   ¦
```

Figure 6. Sample Correlation Matrix

main diagonal is forced to 1.0, despite the fact that round-off errors would produce values slightly off that ideal. Figure 7 is the output using the example data. At this

```
            Y1          Y2          X1          X2

Y1      1.0000      0.5687      0.8445      0.6479
Y2      0.5687      1.0000      0.4728      0.8755
X1      0.8445      0.4728      1.0000      0.5758
X2      0.6479      0.8775      0.5758      1.0000
```

Figure 7. CORRMATRIX Output Example

point, the user is given the option of exiting CANCOR without calculating further canonical correlation statistics.

<u>Calculate CANCOR Statistics</u>. The key to calculating further statistics is the solving of an eigenvalue problem; in this case, the eigenvalues of a product of the partitions

of CORRMATRIX. The procedure used to estimate the polynomial roots (eigenvalues) is called the deflation method. The most popular method for estimating the largest eigenvalue is called the power method. The deflation method uses the power method to determine the largest eigenvalue and eigenvector, factors out those values from the matrix (deflate the matrix), and then reapplies the power method to the deflated matrix. The user has the option of setting the Eigenvalue routine stopping criteria (Epsilon) to any value between 0.1 and 0.000001 inclusive with default at 0.0001.

To be used by CANCOR, it is necessary to generate matrix 'A' by multiplying the partitions of CORRMATRIX:

$$A = (R(YY)^{-1} * R(YX) * R(XX)^{-1} * R(XY))$$

Data multicollinearity is indicated when either of the inverses of two self-correlation matrices is nonexistant. If that occurs, the CANCOR procedure is exited after warning the user. Otherwise, the canonical correlation (CANCOR), Wilk's Lambda, and CHI-Square statistics are calculated from the eigenvalues and then printed. Figure 8 is the output using the example data and the most accurate Epsilon setting. At this point, the user is given the option of exiting CANCOR without proceeding with canonical variate calculations.

Canonical Variate Coefficients. The next routine called by CANCOR calculates the Canonical Variate Coefficients (ALPHA & BETA vectors) for both sets of variables (X & Y)

```
                          Canonical     Wilk's    CHI-
Number   Eigenvalue     Correlation    Lambda    Square

   1       0.8323          0.9123      0.0842   21.0366
   2       0.4980          0.7057      0.5020    5.8580
```

Figure 8. CANCORSTATS Output Example

and prints them. It should be noted that the ALPHA's are the normalized eigenvectors. Calculations are as follows:

$$ALPHA = 1/SQRT(C) * ALPHA$$

where

$$C = ALPHA^{T} * R(YY) * ALPHA$$

and

$$BETA = (1/CANCOR) * (R(XX)^{-1} * R(XY)) * ALPHA$$

where

CANCOR = i'th Canonical Correlation

Figure 9 is the output using the example data.

```
COEFFICIENTS FOR CANONICAL VARIABLES OF THE FIRST SET

                   CANVAR 1   CANVAR 2

          X1        0.5337     1.0923
          Y2        0.5949    -1.0602

COEFFICIENTS FOR CANONICAL VARIABLES OF THE SECOND SET

                   CANVAR 1   CANVAR 2

          X1        0.3821     1.1619
          X2        0.7299    -0.9814
```

Figure 9. Canonical Variate Coefficients Output Example

<u>Canonical Variate Scores</u>. Once the ALPHA and BETA

coefficients have been calculated, CANCOR calls a routine to calculate the Canonical Variate scores. The scores are then printed and/or saved as desired by the user. The scores are stored in a data array the same size as the orignal data. If the user chooses to save the scores, a properly formatted data disk must be in the non-boot drive prior to calling the SAVEFILE routine. If printed, the scores are printed one page at a time in the same manner as in the ECHOFILE routine. Figure 10 is the output using the example data.

```
              CANVAR 1           CANVAR 2
           First   Second     First   Second

  1       -1.761   -1.576    -0.971   -0.876
  2       -1.582   -1.630     0.353    0.579
  3       -0.367    0.005    -0.990   -0.929
  4       -0.902   -0.946     0.786    1.041
  5        0.027    0.109     0.774    1.006
  6        0.563    1.060    -1.003   -0.964
  7        0.957    0.533     0.761   -0.946
  8        1.242    0.636    -0.570    0.988
  9        0.098   -0.679    -0.996   -1.390
 10        0.206    0.266     2.098    1.485
 11        0.277    0.848     0.328    0.012
 12        1.242    1.375    -0.570   -0.006
```

Figure 10. Canonical Variate Scores Output Example

<u>Canonical Loadings</u>. The last routine called by CANCOR calculates and prints the Structure Correlations (canonical loadings) and the Indexes of Redundancy (overlapping information) in the two sets of variables. These values are based on the Canonical Variate Coefficients (ALPHA & BETA), the Eigenvalues, and the Sample Correlation Matrix. Figures 11 and 12 are the outputs using the example data. Once the

```
STRUCTURE CORRELATIONS

             Y1          Y2

YCV1      0.8721      0.4894
YCV2      0.8985     -0.4390

             X1          X2

XCV1      0.8024      0.5968
XCV2      0.9499     -0.3124
```

Figure 11. Canonical Loadings Output Example

```
INDEXES OF REDUNDANCY

    VY1  =  0.6524
    VY2  =  0.1076
            ------
            0.7601 of total variance

    VX1  =  0.6435
    VX2  =  0.1130
            ------
            0.7565 of total variance
```

Figure 12. Indexes of Redundancy Output Example

Indexes are printed, CANCOR is exited and control is returned to the Top Level menu.

III. Factor Analysis

The Factor Analysis Module (FACTOR) starts by having the user designate a set of manifestation variables. The routines in FACTOR then aid the user in looking for an underlying pattern of relationships between members of the designated set of variables so that a possible reduction to a smaller set of factors or components can be done without a significant loss of accuracy. The module produces and outputs the Factor Loadings, Communalities, Coefficients, and Scores for the designated set of variables. The data in Table II is the data used in the examples throughout this section.

X1	X2	X3	X4
1	2	1	2
2	4	3	1
3	6	7	5
4	7	6	3
5	5	4	7
6	8	9	6
7	9	8	7
8	7	10	9
9	10	11	11
10	8	9	8
11	12	11	10
12	9	13	14

Table II. Example FACTOR Data

Assign Variables. When first activated by the user, FACTOR calls ASSIGNVARS for variable selection. The user is then shown the current status for each of the fields and asked whether to assign a manifestation, remove assignment, or exit ASSIGNVARS. Once entered, this routine will not allow the user to exit until at least two variables are assigned. This means that there should be at least two variables in the data base before calling FACTOR. The user has the option of making as many changes as needed to correctly assign the variables.

Calculate Statistics. The next routine called by FACTOR calculates and prints the means and standard deviations for each of the selected variables. If a printer is on-line, the results are printed there as well as on the screen. Figure 13 is the output using the example data. At this point, the user

```
VARIABLE      MEAN     STANDARD DEVIATION

   X1       6.50000        3.60555
   X2       7.25000        2.73446
   X3       7.66667        3.60135
   X4       6.91667        3.82476
```

Figure 13. Statistics Output Example

is given the option of exiting FACTOR without proceeding to data standardization.

Standardize. In preparation for calculating the Sample Correlation Matrix, the selected variables are standardized by subtracting the field mean and dividing by the field standard deviation. It should be noted that if there is only one record, the standard deviations are zero, the data values become undefined, and are represented as 99.9999. The standardized values are written over the original values in the data array, but are not automatically saved to disk.

Generate Correlation Matrix. The next procedure called by FACTOR generates and prints the Sample Correlation Matrix. The matrix is generated using the same partition method as the Correlation Matrix generated in CANCOR; in this case, the manifestation variables are divided into two equal or nearly equal sets and the routine proceeds as before. Figure 14 is the output using the example data. At this point, the user is

	X1	X2	X3	X4
X1	1.0000	0.8529	0.9102	0.9262
X2	0.8529	1.0000	0.8862	0.7497
X3	0.9102	0.8862	1.0000	0.8888
X4	0.9262	0.7497	0.8888	1.0000

Figure 14. CORRMATRIX Output Example

given the option of exiting FACTOR without calculating any factors.

Factor Generation and Selection. The most significant factor (or principal component) is associated with the largest eigenvalue and eigenvector of the matrix just generated. Once the largest is extracted, the next largest eigenvalue and eigenvector are associated with the next most significant factor, and so on. The method used to solve for the eigenvalues and eigenvectors is the same as used by CANCOR. The next routine called by FACTOR calculates and prints the percents of variance explained by each factor and then has the user select the number of factors to maintain for further analysis. Figure 15 is the output using the example data and the most accurate Epsilon setting. The

FACTOR	EIGENVALUE	PCT OF VAR	CUM PCT
1	3.6090	90.2	90.2
2	0.2606	6.5	96.7
3	0.0837	2.1	98.8
4	0.0467	1.2	100.0

Figure 15. Factor Calculation Output Example

user then has the option of factor selection based on default, a Scree test, a Bartlett Sphericity test, or user selection.

Default. If selected by the user, the number of factors maintained is the number of eigenvalues greater than 1.0.

Scree Test. If selected by the user, SCREE generates a plot of Eigenvalue Magnitudes vs. Factor Numbers. The user is asked to visualize a line passing through the right most points and extending to the left. The most significant factors are those that do NOT fall on the line PLUS the first one that does. That is the number of factors that should be kept. The nearly flat aspect of the remaining factors indicates little improvement if more are kept.

Bartlett Sphericity Test. If selected by the user, BARTLETT calculates the CHI-Square statistic for the Bartlett test of significance for as many factors as the user desires.

The user is then asked to select the number of significant factors that should be kept. The test statistic is used to check the hypothesis

```
Ho: EIGVAL(r+1)=EIGVAL(r+2)=...=EIGVAL(k)=0
       vs.
Ha: EIGVAL(r+1) <> 0; after 'r' tests.
```

The user should reference a CHI-Square table for

```
CHI ( a , (k-r)(k-r-1) )
```

where

```
a = significance level
k = number of factors
r = number of tests done
```

and reject the null hypothesis if the test statistic is larger.

The routine will calculate one test statistic, then ask the user if more should be calculated. It is up to the user to decide when to stop. It will stop automatically after making as many calculations as there are eigenvalues. The routine will then call USERSELECT, as described below, to get the number of factors to be kept. This test is good for small samples (n < 100) or for a large number of manifestation variables (k > 9).

<u>User Select</u>. If selected by the user or BARTLETT, USERSELECT asks the user to enter the number of factors to be kept for further analysis. That number must be between 1 and N (number of eigenvalues) inclusive.

<u>Factor Matrix</u>. The next routine called by FACTOR

calculates and prints the factor matrix of Loadings for each factor (N factors), the Communalities based on the number of factors selected (NS selected), and the Factor Score Coefficients for each of the designated manifestation variables under analysis. Figures 16, 17, and 18 are the outputs using the example data and assume 1 factor was retained for analysis.

```
FACTOR MATRIX USING PRINCIPAL FACTOR(S)

                          FACTOR 1

            X1             0.9717
            X2             0.9171
            X3             0.9704
            X4             0.9391
```

Figure 16. Factor Matrix Output Example

```
VARIABLE    COMMUNALITY

   X1          0.9443
   X2          0.8412
   X3          0.9417
   X4          0.8819
```

Figure 17. Variable Communality Output Example

```
FACTOR SCORE COEFFICIENTS

                FACTOR 1

  X1             0.2692
  X2             0.2541
  X3             0.2689
  X4             0.2602
```

Figure 18. Factor Score Coefficients Output Example

Factor Scores. Once the Factor Score Coefficients have been calculated, FACTOR calls a routine to calculate the Factor Scores. The scores are then printed and/or saved as desired by the user. The scores are stored in a data array the same size as the original data. If the user chooses to save the scores, a properly formatted data disk must be in the non-boot drive prior to calling the SAVEFILE routine. If printed, the scores are printed one page at a time in the same manner as in the ECHOFILE routine. Figure 19 is the output using the example data. Once the appropriate option is

```
FACTOR SCORES:

             CASE    FACT 1

                1   -1.7309
                2   -1.3890
                3   -0.5577
                4   -0.6008
                5   -0.5892
                6    0.0696
                7    0.2305
                8    0.4047
                9    0.9689
               10    0.5043
               11    1.2361
               12    1.4535
```

Figure 19. Factor Scores Output Example

completed, FACTOR is exited and control is returned to the Top Level menu.

IV. Formatting Blank Disks

Before using a blank disk to save data files, it is necessary to format it in a form that can be used by the Apple PASCAL operating system. Once formatted, the disk should be marked so that it is not formatted again.

Formatting is done by inserting the disk with side 2 up; either after booting or running the package. Figure 20 outlines the way to format a disk. If the user has inserted

```
>COMMAND: E(DIT, R(UN, F(ILE, C(OMP, L(INK, X(ECUTE...
 X

>EXECUTE WHAT FILE?
 APPLE:FORMATTER     (return)

>APPLE DISK FORMATTER PROGRAM
>FORMAT WHICH DISK (4, 5, 9..12) ?
 5 (return)      (Non-boot drive)

>NOW FORMATTING DISKETTE IN DRIVE 5  (If selected)
  or
>DESTROY DIRECTORY OF BLANK ?  (Disk already formatted)
```

Figure 20. Blank Disk Formatting

a new disk or answered 'Y' to the second response above, the non-boot drive will make some whirring sounds for a few moments, then the 'FORMAT WHICH DISK' statement will appear again. Press the (return) key to return to the Command level again if no more disks are to be formatted. NOTE: If the system was booted with side 1 up, the disk should be turned over prior to pressing the (return).

V. Special Features

There are several special routines available on side 2 in SYSTEM.FILER. While there are 17 different commands available in the FILER as written by Apple, only 5 are discussed here. Caution should be observed by the unsophisticated user because the other commands could contaminate any disks on-line. To exit any routines entered by accident, press the (return) key to return to the command line.

To execute any of the commands in the FILER, insert the disk with side 2 up; either after booting or running the package. Be sure that a preformatted data disk is on-line in the non-boot drive. Figure 21 shows how to execute the FILER.

```
>COMMAND: E(DIT, R(UN, F(ILE, C(OMP, L(INK, X(ECUTE,.
 F

>Filer: G, S, N, L, R, C, T, D, Q (1.1)
 <command>   (Where <command> = L, R, C, K, or Q)
```

Figure 21. SYSTEM.FILER Execution

Once selected, each of these commands carries out a different procedure.

L(IST DIRECTORY. If selected, L(ist works as follows:

```
>Dir listing of ?   BLANK:   (return)
```

If BLANK: is on-line, the FILER will list the names of all data files on the disk, the size of the file in disk segments, and a date. If BLANK: is not found, a message is displayed and the FILER command line will then return.

<u>R(EMOVE</u>. If selected, R(emove works as follows:

```
>Remove ?   BLANK:<filename>.TEXT   (return)
>BLANK:<filename>.TEXT   -->  removed
>Update directory?  (Y/N)
```

If the file is found, FILER repeats the filename to verify that it is the correct one to remove. If the user responds with a 'Y', the FILER will remove the designated filename from the directory and that file is considered erased. (The sophisticated user can recover the file with the M(AKE command.) If the file is not found, a message is displayed and the FILER command line will then return.

<u>C(HANGE</u>. If selected, C(hange works as follows:

```
>Change ?  BLANK:<filename>.TEXT  (return)
>Change to what ?  BLANK:<newfilename>.TEXT  (return)
```

If the file is found, the FILER will change the name of the designated <filename> to <newfilename>. If the file is not found, a message is displayed and the FILER command line will then return. NOTE: The <newfilename> must be 10 or less characters long and start with a letter.

<u>K(RUNCH</u>. If selected, K(runch works as follows:

```
>Crunch ?   BLANK:   (return)
>From end of disk, block 280 ? (Y/N)
```

If BLANK: is on-line and 'Y' was selected, K(runch will move the files forward and all the available space will be at the end of the disk. Typing an 'N' will cause the prompt

```
>Starting at block #
```

requiring the user to input a number from 1 to 280. This option should NOT be used. After BLANK: is krunched, the FILER command line will then return.

Q(UIT. When selected, Q(uit will return control to the top level command line shown above. NOTE: If the system was booted with side 1 up, the disk should be turned over prior to quitting; otherwise side 2 should remain up.

VI. Package Construction

The PASCAL Statistical Procedures Package (PSPP) was written on the Apple IIe microcomputer using the Apple PASCAL language and operating system. It should be executable on the Apple II or Apple II+ (with language card installed), and Apple III computers (if the source is recompiled).

The package is composed of a host program stored in file PSPP.CODE, 4 regular units stored in the file MYLIB.CODE, and 16 intrinsic units stored in the file SYSTEM.LIBRARY. It should be noted that several intrinsic units normally stored in the SYSTEM.LIBRARY were removed because they are not used by PSPP. The Apple PASCAL operating system files SYSTEM.APPLE, SYSTEM.PASCAL, and SYSTEM.MISCINFO are on both sides of the program disk. Operating system files SYSTEM.FILER, FORMATTER.CODE, and FORMATTER.DATA are additionally stored on side 2.

There are a total of 163 new procedures, in addition to those intrinsic procedures by Apple in the SYSTEM.LIBRARY. The text files for these procedures take over 218,000 bytes of storage space. Compiled, they take up almost 100,000 bytes of storage space. By using the PASCAL unit structures, the most core used by the program at any one time is about 21,000 bytes. There are more than 17,000 bytes used for data storage. The maximum user available space in the Apple IIe is 39,900 bytes of core when the SWAPPING option is set. The

Apple PASCAL operating system files necessary to execute the program take up 84 blocks of diskette space. The compiled versions of the host program and the two libraries take up 215 blocks. There are 274 blocks available for use.

The package was put together by first compiling the intrinsic units that did not use any others and storing them in the SYSTEM.LIBRARY. Next, intrinsic units that referenced others were compiled and stored. Regular units that did not use any others were then compiled and stored in MYLIB.CODE. Regular units that referenced others could then be compiled and stored. Lastly, after all units (intrinsic and regular) were compiled and stored in their appropriate libraries, the host program, PSPP, was compiled and linked with the regular units. Because the compiled versions more than filled the available space on a single diskette, both sides were used with the Apple PASCAL operating system files necessary for special non-program features stored only on side 2.

The package structure is outlined in Figure 22. The boxes stand for procedures called by the normal program flow while the ovals are options the user has a choice of in the box just above them in the tree. On the DATA side of the tree, once a procedure is completed, control 'backs up' to a higher level menu until the user is done. On the statistical side, control proceeds toward the end of the procedure with several opportunities to exit to the top level (PSPP) menu.

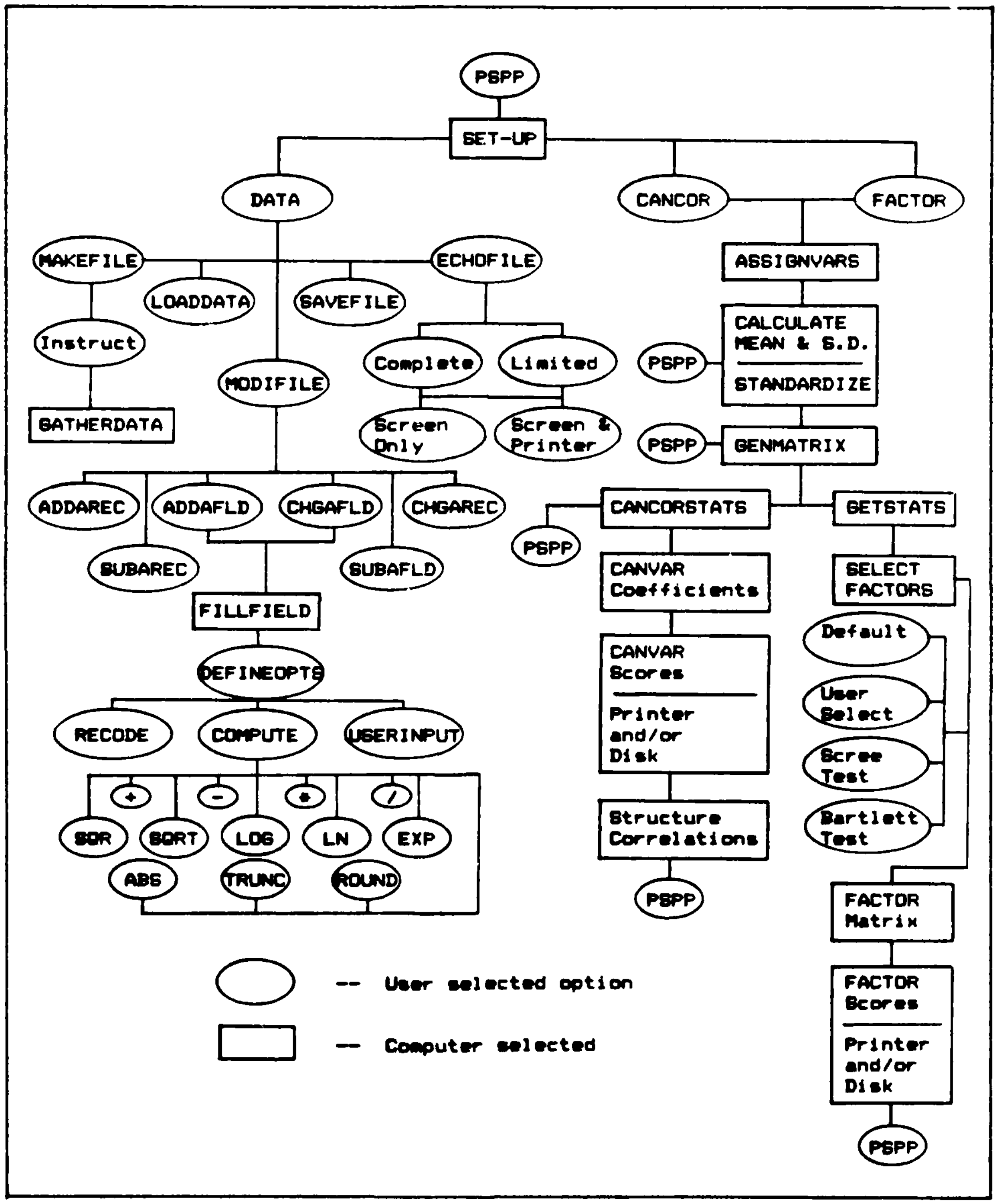


Figure 22. PSPP Package Structure

Appendix B

Validation Runs

SPSS Output of Example CANCOR Data

```
VARIABLE            MEAN             STANDARD DEV

  Y1               6.0000                2.4863
  Y2               5.7500                2.3789
  X1               6.3333                2.4246
  X2               6.4167                1.9752



CORRELATION COEFFICIENTS

             Y1            Y2            X1            X2

  Y1   1.00000        .56869        .84449        .64789
  Y2    .56869       1.00000        .47284        .87546
  X1    .84449        .47284       1.00000        .57579
  X2    .64789        .87546        .57579       1.00000



                             CANONICAL       WILK S
NUMBER    EIGENVALUE       CORRELATION      LAMBDA     CHI-SQUARE

   1        .83232            .91232        .08418      22.27354
   2        .49799            .70568        .50201       6.20223



COEFFICIENTS FOR CANONICAL VARIABLES OF THE FIRST SET

                CANVAR   1     CANVAR   2

  Y1               .53370      -1.09232
  Y2               .59498       1.06019

COEFFICIENTS FOR CANONICAL VARIABLES OF THE SECOND SET

                CANVAR   1     CANVAR   2

  X1               .38209      -1.16189
  X2               .72995        .98140
```

```
CASE          CANVAR  1             CANVAR  2
           FIRST     SECOND      FIRST     SECOND

  1     -1.76107   -1.57596     .97107     .87582
  2     -1.58187   -1.63034   -.35326    -.57944
  3      -.36678     .00546     .99008     .92877
  4      -.90245    -.94561   -.78625   -1.04100
  5       .02707     .10866   -.77358   -1.00570
  6       .56274    1.05974    1.00275     .96407
  7       .95660     .53260   -.76091     .94642
  8      1.24216     .63580     .56976    -.98804
  9       .09798    -.67927     .99642    1.39032
 10       .20627     .26625   -2.09791   -1.48490
 11       .27718     .84777   -.32791    -.01199
 12      1.24216    1.37491     .56976     .00566


STRUCTURE CORRELATIONS

                       Y1        Y2

       YCANVAR1       .8721    -.4894
       YCANVAR2       .8985     .4390

                       X1        X2

       XCANVAR1       .8024    -.5968
       XCANVAR2       .9500     .3124
```

SPSS Output of Example FACTOR Data

```
VARIABLE            MEAN          STANDARD DEV

   X1              6.5000            3.6056
   X2              7.2500            2.7345
   X3              7.6667            3.6013
   X4              6.9167            3.8248


CORRELATION COEFFICIENTS

            X1           X2           X3           X4

   X1   1.00000       .85292       .91015       .92621
   X2    .85292      1.00000       .88622       .74971
   X3    .91015       .88622      1.00000       .88879
   X4    .92621       .74971       .88879      1.00000


FACTOR     EIGENVALUE     PCT OF VAR     CUM PCT

   1        3.60903          90.2          90.2
   2         .26059           6.5          96.7
   3         .08369           2.1          98.8
   4         .04669           1.2         100.0


FACTOR MATRIX USING PRINCIPAL FACTOR

            FACTOR  1

   X1         .97173
   X2         .91714
   X3         .97043
   X4         .93909


VARIABLE    COMMUNALITY

   X1         .94426
   X2         .84115
   X3         .94173
   X4         .88189
```

```
FACTOR SCORE COEFFICIENTS

                FACTOR  1

    X1          .26925
    X2          .25412
    X3          .26889
    X4          .26021
```

CANCOR Validation Data

ECHOCHECK OF CURRENT DATAFILE:

INDEX	Tot I/O	Lines	Hrdwar	TurnAr	Cards	Depart
1	3076.56	10.0000	42.0000	0.06240	43.0000	8.13780
2	1124.72	8.00000	165.000	0.07230	6.00000	8.22710
3	176.868	5.00000	210.000	0.07130	20.0000	8.25340
4	821.637	21.0000	316.000	0.11410	8.00000	8.32830
5	1095.27	11.0000	40.0000	0.15440	18.0000	8.47030
6	1024.81	21.0000	659.000	0.19220	9.00000	8.52730
7	2373.48	9.00000	297.000	0.32410	7.00000	8.62240
8	78.4809	0.00000	251.000	0.06430	4.00000	8.63080
9	138.984	23.0000	59.0000	0.01590	0.00000	8.63890
10	64.8879	28.0000	87.0000	0.03230	1.00000	8.74040
11	4708.77	6.00000	113.000	0.38740	25.0000	8.77780
12	302.434	12.0000	337.000	0.12440	2.00000	8.80590
13	552.740	6.00000	120.000	0.11120	1.00000	8.85430
14	548.733	10.0000	165.000	0.08680	26.0000	8.86470
15	697.457	17.0000	180.000	0.05270	1.00000	8.89840
16	1526.31	9.00000	88.0000	0.13560	24.0000	8.90240
17	3669.65	25.0000	39.0000	0.13680	2.00000	8.95300
18	560.525	10.0000	236.000	0.06200	10.0000	9.20240
19	1486.57	3.00000	78.0000	0.08230	1.00000	9.23920
20	1737.07	6.00000	295.000	0.13950	2.00000	9.24470
21	2086.30	12.0000	179.000	0.09410	2.00000	9.35100
22	73.0646	18.0000	334.000	0.09660	14.0000	9.48360
23	1027.91	16.0000	210.000	0.15030	2.00000	9.49070
24	3314.01	8.00000	194.000	0.22230	10.0000	9.52730
25	414.305	0.00000	465.000	0.20540	9.00000	9.71140
26	656.447	6.00000	283.000	0.33220	1.00000	9.73990
27	407.300	36.0000	179.000	0.19900	2.00000	9.74330
28	769.839	30.0000	225.000	0.33410	1.00000	9.77580
29	877.594	28.0000	186.000	0.15380	3.00000	9.88480
30	1351.18	16.0000	248.000	0.17790	4.00000	9.90780
31	313.948	10.0000	258.000	0.19580	29.0000	9.98560
32	1100.58	2.00000	192.000	0.23920	9.00000	9.99110
33	415.781	1.00000	105.000	0.19770	3.00000	10.0072
34	1880.95	7.00000	144.000	0.35360	1.00000	10.0263
35	587.878	2.00000	134.000	0.16340	11.0000	10.0958
36	428.761	27.0000	131.000	0.14870	1.00000	10.1018
37	1150.05	17.0000	127.000	0.29510	21.0000	10.1448
38	538.216	11.0000	63.0000	0.12970	15.0000	10.1601
39	730.358	9.00000	155.000	0.12390	3.00000	10.2241
40	2598.05	6.00000	111.000	0.35520	27.0000	10.2414
41	160.259	13.0000	55.0000	0.04000	35.0000	10.2512
42	2233.55	0.00000	157.000	0.24120	19.0000	10.2833
43	1825.80	24.0000	70.0000	0.18050	41.0000	10.2990
44-	841.909	0.00000	191.000	0.08900	11.0000	10.3161
45	2220.95	6.00000	86.0000	0.21220	10.0000	10.3362
46	3085.98	15.0000	49.0000	0.10480	37.0000	10.3743
47	881.816	52.0000	261.000	0.04730	3.00000	10.4723
48	2093.65	16.0000	106.000	0.05910	9.00000	10.4832
49	893.343	10.0000	131.000	0.07010	1.00000	10.5797
50	1392.10	2.00000	219.000	0.10310	6.00000	10.5887
51	1629.82	10.0000	255.000	0.09510	2.00000	10.7384
52	544.712	7.00000	53.0000	0.02040	0.00000	10.8251
53	129.012	8.00000	151.000	0.01400	2.00000	10.8997
54	227.130	8.00000	267.000	0.08650	2.00000	11.0006
55	943.319	0.00000	196.000	0.13490	0.00000	11.0388

56	957.987	3.00000	67.0000	0.03860	1.00000	11.0912
57	621.068	53.0000	118.000	0.04880	2.00000	11.1070
58	807.440	17.0000	506.000	0.14880	6.00000	11.2590
59	726.578	6.00000	141.000	0.03370	17.0000	11.2823
60	708.572	13.0000	185.000	0.09640	5.00000	11.3879
61	1099.20	1.00000	375.000	0.16270	7.00000	11.4754
62	86.3832	1.00000	228.000	0.11390	1.00000	11.5361
63	352.511	8.00000	254.000	0.11440	0.00000	11.5965
64	2500.94	3.00000	110.000	0.31540	0.00000	11.6101
65	60.2336	6.00000	37.0000	0.07900	2.00000	11.6272
66	1068.36	14.0000	216.000	0.24110	24.0000	11.6605
67	401.698	2.00000	85.0000	0.08890	2.00000	11.6909
68	2253.70	17.0000	183.000	0.38980	13.0000	11.7870
69	1315.64	4.00000	554.000	0.29940	16.0000	11.8338
70	24.1495	5.00000	166.000	0.14050	1.00000	11.8560
71	1441.28	8.00000	149.000	0.21700	9.00000	11.8758
72	331.660	15.0000	39.0000	0.03300	21.0000	11.8957
73	489.303	5.00000	36.0000	0.14940	3.00000	11.8979
74	2437.51	9.00000	160.000	0.31180	13.0000	11.9229
75	2084.17	7.00000	317.000	0.54090	3.00000	11.9672
76	785.460	0.00000	425.000	0.12460	2.00000	12.0679
77	975.113	3.00000	69.0000	0.09610	2.00000	12.0817
78	2214.27	1.00000	133.000	0.13680	1.00000	12.1065
79	887.103	24.0000	326.000	0.16290	1.00000	12.1638
80	3840.95	17.0000	74.0000	0.13730	17.0000	12.1852
81	1932.12	1.00000	101.000	0.07970	16.0000	12.2046
82	1157.38	15.0000	257.000	0.09400	7.00000	12.3195
83	565.000	9.00000	95.0000	0.06440	15.0000	12.3717
84	1991.76	5.00000	265.000	0.15520	4.00000	12.5238
85	2028.79	2.00000	448.000	0.23870	19.0000	12.5722
86	3625.62	6.00000	106.000	0.31530	1.00000	12.5908
87	780.934	8.00000	271.000	0.19400	1.00000	12.6297
88	1083.73	1.00000	86.0000	0.22600	25.0000	12.6721
89	1497.49	6.00000	139.000	0.17920	0.00000	12.6905
90	4096.66	20.0000	188.000	0.35350	3.00000	12.7302
91	234.093	7.00000	150.000	0.04440	2.00000	12.7747
92	1175.53	5.00000	238.000	0.13250	0.00000	12.8360
93	2613.84	15.0000	309.000	0.30950	7.00000	12.8512
94	31.8972	12.0000	270.000	0.15770	11.0000	12.9304
95	332.294	7.00000	214.000	0.28920	3.00000	13.0122
96	518.027	8.00000	455.000	0.24120	11.0000	13.0830
97	1054.52	28.0000	110.000	0.27800	9.00000	13.0968
98	965.087	5.00000	121.000	0.26050	10.0000	13.1320
99	1356.21	2.00000	99.0000	0.12720	17.0000	13.2084
100	1579.61	23.0000	249.000	0.28020	25.0000	13.2470
101	159.325	5.00000	340.000	0.15920	0.00000	13.3175
102	801.515	1.00000	193.000	0.24350	5.00000	13.4075
103	979.642	15.0000	265.000	0.36870	11.0000	13.4154
104	2848.71	3.00000	109.000	0.33180	3.00000	13.4343
105	3328.76	0.00000	349.000	0.50170	35.0000	13.4853
106	1198.12	13.0000	104.000	0.27900	1.00000	13.4918
107	305.168	10.0000	155.000	0.05070	0.00000	13.5436
108	546.161	3.00000	266.000	0.31100	3.00000	13.5614
109	2388.16	2.00000	320.000	0.47360	1.00000	13.5750
110	1526.71	25.0000	99.0000	0.11320	5.00000	13.6179
111	4117.59	6.00000	91.0000	0.38960	13.0000	13.6457
112	2954.08	2.00000	280.000	0.43960	2.00000	13.7259
113	936.728	10.0000	364.000	0.22530	7.00000	13.8910
114	630.684	1.00000	613.000	0.21190	13.0000	13.9461
115	1635.68	17.0000	36.0000	0.29440	1.00000	13.9545

116	1358.49	5.00000	265.000	0.38900	4.00000	13.9767
117	942.427	4.00000	172.000	0.20300	32.0000	14.0090
118	793.553	6.00000	227.000	0.19140	5.00000	14.0731
119	1556.48	26.0000	25.0000	0.22690	9.00000	14.0886
120	166.166	28.0000	86.0000	0.20830	6.00000	14.1123
121	1957.00	22.0000	91.0000	0.36760	14.0000	14.1182
122	21.0423	1.00000	99.0000	0.04930	1.00000	14.1471
123	1039.40	21.0000	243.000	0.27790	29.0000	14.1672
124	986.860	45.0000	257.000	0.14240	2.00000	14.2621
125	2093.50	7.00000	209.000	0.20760	6.00000	14.2643
126	2525.62	9.00000	56.0000	0.27270	7.00000	14.2747
127	2678.46	7.00000	70.0000	0.17010	10.0000	14.2871
128	500.107	12.0000	174.000	0.14260	33.0000	14.2932
129	1178.48	6.00000	57.0000	0.04850	22.0000	14.4148
130	850.081	8.00000	158.000	0.06910	6.00000	14.4226
131	633.805	16.0000	122.000	0.05870	3.00000	14.5275
132	252.776	11.0000	203.000	0.07210	12.0000	14.5864
133	1742.68	14.0000	74.0000	0.13490	4.00000	14.6158
134	492.352	0.00000	34.0000	0.04150	23.0000	14.6682
135	1229.26	10.0000	183.000	0.16120	26.0000	14.6750
136	756.081	0.00000	159.000	0.09050	4.00000	14.7067
137	948.293	7.00000	172.000	0.10780	16.0000	14.7271
138	2194.86	9.00000	239.000	0.23100	22.0000	14.7430
139	21.9734	5.00000	32.0000	0.00760	12.0000	14.8541
140	2979.86	2.00000	157.000	0.08980	22.0000	14.8971
141	169.686	22.0000	147.000	0.01200	15.0000	14.9349
142	810.245	24.0000	21.0000	0.03280	2.00000	15.0619
143	1881.51	8.00000	288.000	0.12370	19.0000	15.1005
144	864.776	9.00000	63.0000	0.03260	4.00000	15.1826
145	375.984	0.00000	103.000	0.05190	2.00000	15.2528
146	732.635	17.0000	102.000	0.07070	8.00000	15.2609
147	1632.64	18.0000	237.000	0.13790	9.00000	15.2991
148	1584.56	6.00000	538.000	0.15660	1.00000	15.4588
149	2300.12	31.0000	118.000	0.12120	3.00000	15.5085
150	2176.96	5.00000	200.000	0.09160	45.0000	15.5348
151	7283.18	3.00000	56.0000	0.31900	12.0000	15.5486
152	814.411	3.00000	39.0000	0.02480	4.00000	15.6460
153	93.9386	2.00000	389.000	0.13420	13.0000	15.7924
154	492.445	0.00000	71.0000	0.29800	10.0000	15.9632
155	374.711	2.00000	350.000	0.24340	11.0000	15.9933
156	716.375	3.00000	554.000	0.40230	27.0000	16.0481
157	119.654	28.0000	125.000	0.25040	16.0000	16.0857
158	769.155	1.00000	290.000	0.38260	57.0000	16.1398
159	627.935	13.0000	67.0000	0.16080	7.00000	16.1454
160	1204.00	2.00000	38.0000	0.35520	8.00000	16.1554
161	743.828	10.0000	270.000	0.38860	30.0000	16.1835
162	4369.69	6.00000	58.0000	0.55960	19.0000	16.2295
163	2717.28	26.0000	403.000	0.34050	9.00000	16.3157

CANONICAL CORRELATION

```
FILE bigcancor:

 VARIABLE       MEAN      STANDARD DEVIATION

  Tot I/O  1287.09         1104.71
    Lines    10.6319          9.77080
   TurnAr     0.18021         0.11840
   Hrdwar   188.626         125.494
    Cards    10.2577         10.6890
   Depart    12.2921          2.28627


CORRELATION COEFFICIENTS:

               Tot I/O      Lines     TurnAr     Hrdwar      Cards     Depart

  Tot I/O       1.0000    -0.0577     0.4789    -0.1257     0.1686     0.0717
    Lines      -0.0577     1.0000    -0.1062    -0.0751    -0.0951    -0.0983
   TurnAr       0.4789    -0.1062     1.0000     0.2505     0.1543     0.2181
   Hrdwar      -0.1257    -0.0751     0.2505     1.0000    -0.0114     0.0022
    Cards       0.1686    -0.0951     0.1543    -0.0114     1.0000     0.1240
   Depart       0.0717    -0.0983     0.2181     0.0022     0.1240     1.0000


                      CANONICAL    WILK'S    CHI-
NUMBER  EIGENVALUE  CORRELATION   LAMBDA    SQUARE

  1        0.1835      0.4283     0.7766   40.0753
  2        0.0489      0.2211     0.9511    7.9484
  3        0.0000      0.0023     1.0000    0.0008


COEFFICIENTS FOR CANONICAL VARIABLES OF THE FIRST SET

                 CANVAR 1  CANVAR 2  CANVAR 3

       Tot I/O    0.6536    0.9152    0.1816
         Lines    0.2116   -0.3253    0.9278
        TurnAr   -1.0833    0.0230    0.3662

COEFFICIENTS FOR CANONICAL VARIABLES OF THE SECOND SET

                 CANVAR 1  CANVAR 2  CANVAR 3

        Hrdwar   -0.8630   -0.3754   -0.3383
         Cards   -0.1311    0.8041   -0.5935
        Depart   -0.4727    0.3651    0.8118


STRUCTURE CORRELATIONS:

         Tot I/O    Lines    TurnAr

YCV1      0.1226   0.9450   0.3034
YCV2      0.2889  -0.3806   0.8785
YCV3     -0.7927   0.4958   0.3546
```

```
          Hrdwar    Cards   Depart

XCV1    -0.8625  -0.3838  -0.3298
XCV2    -0.1798   0.8537  -0.4889
XCV3    -0.4909   0.4640   0.7375


INDEXES OF REDUNDANCY:

        VY1  =   0.0445
        VY2  =   0.0209
        VY3  =   0.0000
                 -------
                 0.0654 of total variance

        VX1  =   0.0622
        VX2  =   0.0176
        VX3  =   0.0000
                 -------
                 0.0800 of total variance
```

SPSS Output of CANCOR Validation Data

VARIABLE	MEAN	STANDARD DEV
Tot I/O	1287.0926	1104.7059
Lines	10.6319	9.7708
TurnAr	.1802	.1184
Hrdwar	188.6258	125.4944
Cards	10.2577	10.6890
Depart	12.2921	2.2863

CORRELATION COEFFICIENTS

	Tot I/O	Lines	TurnAr	Hrdwar	Cards	Depart
TotI/O	1.00000	-.05771	.47888	-.12574	.16862	.07170
Lines	-.05771	1.00000	-.10617	-.07507	-.09507	-.09832
TurnAr	.47888	-.10617	1.00000	.25053	.15427	.21815
Hrdwar	-.12574	-.07507	.25053	1.00000	-.01145	.00216
Cards	.16862	-.09507	.15427	-.01145	1.00000	.12400
Depart	.07170	-.09832	.21815	.00216	.12400	1.00000

NUMBER	EIGENVALUE	CANONICAL CORRELATION	WILK S LAMBDA	CHI-SQUARE
1	.18347	.42834	.77659	40.20171
2	.04891	.22115	.95109	7.97344
3	.00001	.00227	.99999	.00082

COEFFICIENTS FOR CANONICAL VARIABLES OF THE FIRST SET

	CANVAR 1	CANVAR 2	CANVAR 3
Tot I/O	-.65356	.91517	.18156
Lines	-.21157	-.32532	.92783
TurnAr	1.08327	.02300	.36616

COEFFICIENTS FOR CANONICAL VARIABLES OF THE SECOND SET

	CANVAR 1	CANVAR 2	CANVAR 3
Hrdwar	.86301	-.37543	-.33825
Cards	.13108	.80413	-.59326
Depart	.47275	.36511	.81168

FACTOR Validation Data

ECHOCHECK OF CURRENT DATAFILE:

INDEX	Disk I/O	Arrive	CPU Used	Cards	I/O Time	Lines
1	2886.00	8.07540	12.0000	43.0000	190.564	10.0000
2	1057.00	8.15480	38.0000	6.00000	67.7181	8.00000
3	164.000	8.18210	182.000	20.0000	12.8682	5.00000
4	769.000	8.21420	202.000	8.00000	52.6366	21.0000
5	1011.00	8.31590	11.0000	18.0000	84.2657	11.0000
6	997.000	8.33510	552.000	9.00000	27.8084	21.0000
7	2192.00	8.29830	185.000	7.00000	181.482	9.00000
8	72.0000	8.56650	187.000	4.00000	6.48090	0.00000
9	127.000	8.62300	4.00000	0.00000	11.9842	23.0000
10	59.0000	8.70810	17.0000	1.00000	5.88790	28.0000
11	4438.00	8.39040	5.00000	25.0000	270.773	6.00000
12	287.000	8.68150	306.000	2.00000	15.4339	12.0000
13	507.000	8.74310	61.0000	1.00000	45.7403	6.00000
14	526.000	8.77790	104.000	26.0000	22.7329	10.0000
15	690.000	8.84570	77.0000	1.00000	7.45690	17.0000
16	1421.00	8.76680	54.0000	24.0000	105.310	9.00000
17	3417.00	8.81620	28.0000	2.00000	252.654	25.0000
18	516.000	9.14040	101.000	10.0000	44.5252	10.0000
19	1398.00	9.15690	61.0000	1.00000	88.5690	3.00000
20	1636.00	9.10520	228.000	2.00000	101.066	6.00000
21	1937.00	9.25690	144.000	2.00000	149.300	12.0000
22	68.0000	9.38700	266.000	14.0000	5.06460	18.0000
23	951.000	9.34040	77.0000	2.00000	76.9112	16.0000
24	3145.00	9.30500	87.0000	10.0000	169.014	8.00000
25	390.000	9.50600	320.000	9.00000	24.3051	0.00000
26	615.000	9.40770	269.000	1.00000	41.4471	6.00000
27	380.000	9.54430	117.000	2.00000	27.2997	36.0000
28	713.000	9.44170	163.000	1.00000	56.8394	30.0000
29	803.000	9.73100	152.000	3.00000	74.5937	28.0000
30	1328.00	9.72990	109.000	4.00000	23.1789	16.0000
31	287.000	9.78980	194.000	29.0000	26.9482	10.0000
32	1009.00	9.75190	137.000	9.00000	91.5813	2.00000
33	381.000	9.80950	52.0000	3.00000	34.7807	1.00000
34	1775.00	9.67270	126.000	1.00000	105.954	7.00000
35	538.000	9.93240	76.0000	11.0000	49.8777	2.00000
36	392.000	9.95310	104.000	1.00000	36.7613	27.0000
37	1053.00	9.84970	93.0000	21.0000	97.0487	17.0000
38	494.000	10.0304	44.0000	15.0000	44.2162	11.0000
39	668.000	10.1002	127.000	3.00000	62.3580	9.00000
40	2489.00	9.88620	98.0000	27.0000	109.049	6.00000
41	149.000	10.2112	5.00000	35.0000	11.2590	13.0000
42	2124.00	10.0421	144.000	19.0000	109.550	0.00000
43	1699.00	10.1185	50.0000	41.0000	126.804	24.0000
44	811.000	10.2271	127.000	11.0000	30.9088	0.00000
45	2061.00	10.1240	17.0000	10.0000	159.947	6.00000
46	2887.00	10.2695	35.0000	37.0000	198.979	15.0000
47	806.000	10.4250	67.0000	3.00000	75.8159	52.0000
48	1985.00	10.4241	39.0000	9.00000	108.652	16.0000
49	838.000	10.5096	113.000	1.00000	55.3429	10.0000
50	1306.00	10.4856	185.000	6.00000	86.1001	2.00000
51	1529.00	10.6433	222.000	2.00000	100.823	10.0000
52	511.000	10.8047	20.0000	0.00000	33.7118	7.00000
53	121.000	10.8857	23.0000	2.00000	8.01190	8.00000
54	213.000	10.9141	256.000	2.00000	14.1296	8.00000
55	885.000	10.9039	143.000	0.00000	58.3191	0.00000

56	909.000	11.0526	4.00000	1.00000	48.9866	3.00000
57	572.000	11.0582	92.0000	2.00000	49.0685	53.0000
58	757.000	11.1102	453.000	6.00000	50.4402	17.0000
59	682.000	11.2486	32.0000	17.0000	44.5776	6.00000
60	683.000	11.2915	120.000	5.00000	25.5715	13.0000
61	1019.00	11.3127	342.000	7.00000	80.1963	1.00000
62	79.0000	11.4222	177.000	1.00000	7.38320	1.00000
63	335.000	11.4821	113.000	0.00000	17.5113	8.00000
64	2363.00	11.2947	53.0000	0.00000	137.943	3.00000
65	55.0000	11.5482	2.00000	2.00000	5.23360	6.00000
66	995.000	11.4194	152.000	24.0000	73.3554	14.0000
67	384.000	11.6020	29.0000	2.00000	17.6978	2.00000
68	2101.00	11.3972	46.0000	13.0000	152.703	17.0000
69	1213.00	11.5344	503.000	16.0000	102.637	4.00000
70	22.0000	11.7155	62.0000	1.00000	2.14950	5.00000
71	1407.00	11.6588	97.0000	9.00000	34.2828	8.00000
72	328.000	11.8627	25.0000	21.0000	3.66030	15.0000
73	448.000	11.7485	9.00000	3.00000	41.3030	5.00000
74	2265.00	11.6111	97.0000	13.0000	172.512	9.00000
75	1930.00	11.4263	122.000	3.00000	154.174	7.00000
76	755.000	11.9433	308.000	2.00000	30.4599	0.00000
77	909.000	11.9856	49.0000	2.00000	66.1125	3.00000
78	2058.00	11.9697	18.0000	1.00000	156.268	1.00000
79	857.000	12.0009	135.000	1.00000	30.1031	24.0000
80	3600.00	12.0479	44.0000	17.0000	240.949	17.0000
81	1784.00	12.1249	44.0000	16.0000	148.118	1.00000
82	1075.00	12.2255	205.000	7.00000	82.3815	15.0000
83	521.000	12.3073	85.0000	15.0000	44.0003	9.00000
84	1967.00	12.3686	152.000	4.00000	24.7622	5.00000
85	1890.00	12.3335	396.000	19.0000	138.789	2.00000
86	3383.00	12.2755	94.0000	1.00000	242.621	6.00000
87	720.000	12.4357	203.000	1.00000	60.9340	8.00000
88	996.000	12.4461	28.0000	25.0000	87.7271	1.00000
89	1460.00	12.5113	111.000	0.00000	37.4889	6.00000
90	3783.00	12.3767	49.0000	3.00000	313.665	20.0000
91	214.000	12.7303	93.0000	2.00000	20.0935	7.00000
92	1122.00	12.7035	99.0000	0.00000	33.5250	5.00000
93	2432.00	12.5417	293.000	7.00000	181.836	15.0000
94	29.0000	12.7727	165.000	11.0000	2.89720	12.0000
95	306.000	12.7230	93.0000	3.00000	26.2938	7.00000
96	476.000	12.8418	435.000	11.0000	42.0271	8.00000
97	1020.00	12.8188	46.0000	9.00000	34.5162	28.0000
98	899.000	12.8715	56.0000	10.0000	66.0875	5.00000
99	1256.00	13.0812	83.0000	17.0000	100.207	2.00000
100	1450.00	12.9668	179.000	25.0000	129.615	23.0000
101	146.000	13.1583	284.000	0.00000	13.3249	5.00000
102	733.000	13.1640	142.000	5.00000	68.5154	1.00000
103	898.000	13.0467	122.000	11.0000	81.6425	15.0000
104	2705.00	13.1025	45.0000	3.00000	143.712	3.00000
105	3137.00	12.9836	220.000	35.0000	191.759	0.00000
106	1096.00	13.2128	38.0000	1.00000	102.121	13.0000
107	279.000	13.4929	89.0000	0.00000	26.1682	10.0000
108	500.000	13.2504	69.0000	3.00000	46.1610	3.00000
109	2338.00	13.1014	176.000	1.00000	50.1640	2.00000
110	1438.00	13.5047	67.0000	5.00000	88.7124	25.0000
111	3842.00	13.2561	37.0000	13.0000	275.593	6.00000
112	2754.00	13.2863	162.000	2.00000	200.084	2.00000
113	894.000	13.6657	344.000	7.00000	42.7280	10.0000
114	599.000	13.7342	468.000	13.0000	31.6836	1.00000
115	1531.00	13.6601	5.00000	1.00000	104.676	17.0000

116	1270.00	13.5877	65.0000	4.00000	88.4934	5.00000
117	902.000	13.8060	54.0000	32.0000	40.4272	4.00000
118	745.000	13.8817	175.000	5.00000	48.5530	6.00000
119	1465.00	13.8617	14.0000	9.00000	91.4798	26.0000
120	152.000	13.9040	71.0000	6.00000	14.1660	28.0000
121	1823.00	13.7506	62.0000	14.0000	134.001	22.0000
122	20.0000	14.0078	68.0000	1.00000	1.04230	1.00000
123	967.000	13.8993	134.000	29.0000	72.4030	21.0000
124	902.000	14.1197	237.000	2.00000	84.8598	45.0000
125	1916.00	14.0567	4.00000	6.00000	177.496	7.00000
126	2336.00	14.0020	38.0000	7.00000	189.623	9.00000
127	2480.00	14.1170	14.0000	10.0000	198.461	7.00000
128	476.000	14.1506	42.0000	33.0000	24.1066	12.0000
129	1120.00	14.3663	6.00000	22.0000	58.4752	6.00000
130	783.000	14.3535	139.000	6.00000	67.0806	8.00000
131	584.000	14.4688	94.0000	3.00000	49.8054	16.0000
132	231.000	14.5143	146.000	12.0000	21.7757	11.0000
133	1653.00	14.4809	43.0000	4.00000	89.6848	14.0000
134	487.000	14.6267	2.00000	23.0000	5.35210	0.00000
135	1125.00	14.5138	115.000	26.0000	104.263	10.0000
136	718.000	14.6162	140.000	4.00000	38.0814	0.00000
137	889.000	14.6193	47.0000	16.0000	59.2932	7.00000
138	2052.00	14.5120	171.000	22.0000	142.860	9.00000
139	20.0000	14.8465	5.00000	12.0000	1.97340	5.00000
140	2796.00	14.8073	92.0000	22.0000	183.857	2.00000
141	159.000	14.9229	11.0000	15.0000	10.6861	22.0000
142	770.000	15.0291	3.00000	2.00000	40.2450	24.0000
143	1755.00	14.9768	275.000	19.0000	126.513	8.00000
144	818.000	15.1500	30.0000	4.00000	46.7765	9.00000
145	344.000	15.2009	86.0000	2.00000	31.9842	0.00000
146	687.000	15.1902	46.0000	8.00000	45.6345	17.0000
147	1524.00	15.1612	181.000	9.00000	108.645	18.0000
148	1496.00	15.3022	419.000	1.00000	88.5610	6.00000
149	2138.00	15.3873	50.0000	3.00000	162.120	31.0000
150	2049.00	15.4432	133.000	45.0000	127.956	5.00000
151	6822.00	15.2296	38.0000	12.0000	461.179	3.00000
152	764.000	15.6212	20.0000	4.00000	50.4114	3.00000
153	89.0000	15.6582	326.000	13.0000	4.93860	2.00000
154	457.000	15.6652	4.00000	10.0000	35.4450	0.00000
155	355.000	15.7499	332.000	11.0000	19.7112	2.00000
156	658.000	15.6458	523.000	27.0000	58.3750	3.00000
157	109.000	15.8353	74.0000	16.0000	10.6542	28.0000
158	729.000	15.7572	272.000	57.0000	40.1555	1.00000
159	587.000	15.9846	11.0000	7.00000	40.9350	13.0000
160	1121.00	15.8002	10.0000	8.00000	83.0028	2.00000
161	697.000	15.7949	73.0000	30.0000	46.8279	10.0000
162	4115.00	15.6699	27.0000	19.0000	254.693	6.00000
163	2529.00	15.9752	263.000	9.00000	188.277	26.0000

FACTOR ANALYSIS

```
FILE bigfactor:

 VARIABLE       MEAN      STANDARD DEVIATION

 Disk I/O  1206.10         1035.80
   Arrive    12.1119          2.26339
 CPU Used   123.080        115.763
    Cards    10.2577        10.6890
 I/O Time    80.9944        72.2710
    Lines    10.6319         9.77080


CORRELATION COEFFICIENTS:

               Disk I/O     Arrive    CPU Used      Cards    I/O Time      Lines

 Disk I/O       1.0000      0.0464     -0.1341     0.1685     0.9502     -0.0608
   Arrive       0.0464      1.0000     -0.0065     0.1172     0.0590     -0.0938
 CPU Used      -0.1341     -0.0065      1.0000     0.0132    -0.1415     -0.0923
    Cards       0.1685      0.1172      0.0132     1.0000     0.1629     -0.0951
 I/O Time       0.9502      0.0590     -0.1415     0.1629     1.0000     -0.0105
    Lines      -0.0608     -0.0938     -0.0923    -0.0951    -0.0105      1.0000


FACTOR  EIGENVALUE   PCT OF VAR   CUM PCT

   1      2.0515       34.2        34.2
   2      1.2007       20.0        54.2
   3      0.9769       16.3        70.5
   4      0.8771       14.6        85.1
   5      0.8455       14.1        99.2
   6      0.0483        0.8       100.0


3 factor(s) chosen to continue  FACTOR analysis with.
This explains 70.5% of the variance.


FACTOR MATRIX USING PRINCIPAL FACTOR(S):

                    FACTOR 1    FACTOR 2    FACTOR 3

      Disk I/O       0.9651      0.0820      0.1588
        Arrive       0.1417     -0.5357     -0.6534
      CPU Used      -0.2422     -0.4529      0.6481
         Cards       0.3243     -0.4987     -0.1614
      I/O Time       0.9638      0.1105      0.1323
         Lines      -0.0860      0.6640     -0.2475

        VARIABLE    COMMUNALITY

      Disk I/O       0.9633
        Arrive       0.7340
      CPU Used       0.6837
         Cards       0.3800
      I/O Time       0.9586
         Lines       0.5096
```

```
FACTOR SCORE COEFFICIENTS:

                       FACTOR 1    FACTOR 2    FACTOR 3

          Disk I/O      0.4704      0.0683      0.1625
            Arrive      0.0691     -0.4462     -0.6688
          CPU Used     -0.1180     -0.3772      0.6634
             Cards      0.1581     -0.4154     -0.1652
          I/O Time      0.4698      0.0920      0.1354
             Lines     -0.0419      0.5530     -0.2534
```

SPSS Output of FACTOR Validation Data

VARIABLE	MEAN	STANDARD DEV
Disk I/O	1206.0982	1035.8035
Arrive	12.1119	2.2634
CPU Used	123.0798	115.7630
Cards	10.2577	10.6890
I/O Time	80.9944	72.2711
Lines	10.6319	9.7708

CORRELATION COEFFICIENTS

	Disk I/O	Arrive	CPU Used	Cards	I/O Time	Lines
Disk I/O	1.00000	.04641	-.13413	.16847	.95021	-.06081
Arrive	.04641	1.00000	-.00655	.11718	.05897	-.09376
CPU Used	-.13413	-.00655	1.00000	.01317	-.14151	-.09227
Cards	.16847	.11718	.01317	1.00000	.16294	-.09507
I/O Time	.95021	.05897	-.14151	.16294	1.00000	-.01050
Lines	-.06081	-.09376	-.09227	-.09507	-.01050	1.00000

FACTOR	EIGENVALUE	PCT OF VAR	CUM PCT
1	2.05151	34.2	34.2
2	1.20067	20.0	54.2
3	.97675	16.3	70.5
4	.87709	14.6	85.1
5	.84568	14.1	99.2
6	.04829	.8	100.0

FACTOR MATRIX USING PRINCIPAL FACTOR(S)

	FACTOR 1	FACTOR 2	FACTOR 3
Disk I/O	.96501	-.08189	-.15874
Arrive	.14199	.53534	.65344
CPU Used	-.24205	.45322	-.64814
Cards	.32453	.49866	.16117
I/O Time	.96373	-.11038	-.13229
Lines	-.08620	-.66417	.24697

VARIABLE	COMMUNALITY
Disk I/O	.96314
Arrive	.73374
CPU Used	.68409
Cards	.37996
I/O Time	.95846
Lines	.50955

FACTOR SCORE COEFFICIENTS

	FACTOR 1	FACTOR 2	FACTOR 3
Disk I/O	.47039	-.06820	-.16252
Arrive	.06921	.44587	.66899
CPU Used	-.11799	.37747	-.66357
Cards	.15819	.41532	.16501
I/O Time	.46977	-.09193	-.13544
Lines	-.04202	-.55317	.25285

Appendix C

PSPP

Program Code

```
(*$S+*)

PROGRAM PSPF;

    USES
        TRANSCEND, APPLESTUFF, MAIN_UNIT, MU_A, MU_B, MU_C, MU_D,
                                          MU_E, MU_F, MU_G, MU_H,
        (* Units in SYSTEM.LIBRARY *)     MU_I, MU_J, MU_K,

        (*$U PSPF:MYLIB.CODE *) SET_UP, DATA_MOD,
                                CANCOR_MOD, FACTOR_MOD;

(*************************************************************)
(*                    Internal Procedure                    *)
(*************************************************************)

    PROCEDURE TOPMENU(PRINTER:BOOLEAN);

(*************************************************************)
(*                                                           *)
(*       This procedure displays a menu of user options      *)
(*            and calls the desired statistical module.      *)
(*                                                           *)
(*************************************************************)

        VAR
            OPT:      CHAR;           (* Statistical option to run *)
            DONE:     BOOLEAN;        (* Exit PSPF designator      *)

        BEGIN
        (*$R MAIN_UNIT *)             (* Retain UNIT in memory     *)

            DONE:=FALSE;

            WHILE NOT(DONE) DO
                BEGIN
                    WRITELN(CHR(12),' ':16,CHR(15),' PASCAL ',
                            'STATISTICAL PROCEDURES PACKAGE ',
                            CHR(14));
                    GOTOXY(0,10);

                    WRITELN('Select desired module:',CHR(13));
                    WRITELN('       1 - Data File Preparation');
                    WRITELN('       2 - Canonical Correlation');
                    WRITELN('       3 - Factor Analysis');
                    WRITELN('       4 - Exit PSPF');
                    GETOPTION(OPT);
                    WHILE (OPT<'1') OR (OPT>'4') DO
                        GETOPTION(OPT);

                    CASE (OPT) OF     (* Call appropriate module   *)

                       '1':  DATAMODULE(DATA,SPECS1,SPECS2,PRINTER);
                       '2':      CANCOR(DATA,SPECS1,SPECS2,PRINTER);
                       '3':      FACTOR(DATA,SPECS1,SPECS2,PRINTER);
                       '4':  DONE:=TRUE;

                    END;  (* End of CASE *)
                END;  (* End of WHILE loop *)
        END;  (* End of TOP MENU *)
```

```
(******************************************************************)
(*                       Main body of PSPP                        *)
(******************************************************************)

   BEGIN
   (*$N+*)                                   (* UNIT no-load option *)

      STARTUP(PRINTER);                      (* Display cover       *)

      IF (PRINTER) THEN
         REWRITE(PTR,'PRINTER:');            (* Turn on printer     *)

      TOPMENU(PRINTER);                      (* Select module       *)

      WRITE(CHR(12));
      GOTOXY(28,13);
      WRITE('Done at last. . . !   ');
   END.

(******************************************************************)
```

```
(*$S++)

UNIT SET_UP;

INTERFACE
    USES APPLESTUFF, MAIN_UNIT;

    PROCEDURE STARTUP(VAR PRINTER:BOOLEAN);

IMPLEMENTATION

(*****************************************************************)
(*                        Main body of UNIT                      *)
(*****************************************************************)

PROCEDURE STARTUP;

(*****************************************************************)
(*                                                               *)
(*          This procedure displays a cover, gives a program    *)
(*               overview if desired, then requests the user    *)
(*               input the limits on the size of the data file. *)
(*                                                               *)
(*****************************************************************)

    VAR OPT:  CHAR;                    (* Menu option            *)

(*****************************************************************)
(*                        Internal Procedures                    *)
(*****************************************************************)

    PROCEDURE DRAWSCREEN;

        VAR
            I,                         (* Iteration counter      *)
            POS,                       (* Either X or Y position *)
            PITCH:                     (* Pitch of musical note  *)
                    INTEGER;
            LINE:                      (* Line of text to display *)
                    STRING;

(*****************************************************************)
(*                  Procedures internal to DRAWSCREEN            *)
(*****************************************************************)

        PROCEDURE DISPLAY1;
            BEGIN
                FOR I:=1 TO LENGTH(LINE) DO
                    BEGIN
                        WRITE(COPY(LINE,I,1),' ');
                        NOTE(PITCH,10);
                        PITCH:=PITCH+1;
                    END;
            END;

(*****************************************************************)

        PROCEDURE DISPLAY2;
            BEGIN
                FOR I:=1 TO LENGTH(LINE) DO
```

```
                    BEGIN
                         WRITE(COPY(LINE,I,1),' ');
                         NOTE(PITCH,10);
                         PITCH:=PITCH-1;
                    END;
          END;

(****************************************************************)

     PROCEDURE DISPLAY3;
          BEGIN
               FOR I:=1 TO LENGTH(LINE) DO
                    BEGIN
                         WRITE(COPY(LINE,I,1));
                         NOTE(PITCH,10);
                         PITCH:=PITCH-1;
                    END;
          END;

(****************************************************************)
(*                     Main body of DRAW SCREEN                 *)
(****************************************************************)

     BEGIN
     (*$R APPLESTUFF *)                  (* Retain UNIT in memory    *)
          WRITELN(CHR(12));
          PITCH:=1;

          POS:=1;                        (* Draw box around screen   *)
          FOR I:=1 TO 20 DO
               BEGIN
                    GOTOXY(POS,0);
                    WRITE('*');
                    POS:=POS+4;
               END;

          POS:=2;
          FOR I:=1 TO 10 DO
               BEGIN
                    GOTOXY(77,POS);
                    WRITE('*');
                    POS:=POS+2;
               END;

          POS:=73;
          FOR I:=1 TO 19 DO
               BEGIN
                    GOTOXY(POS,20);
                    WRITE('*');
                    POS:=POS-4;
               END;

          POS:=18;
          FOR I:=1 TO 9 DO
               BEGIN
                    GOTOXY(1,POS);
                    WRITE('*');
                    POS:=POS-2;
               END;
```

```
        WRITE(CHR(15));                  GOTOXY(21,3);    WRITE(' ');
        LINE:='PASCAL STATISTICAL';      GOTOXY(22,3);    DISPLAY1;
                                         GOTOXY(21,5);    WRITE(' ');
        LINE:='PROCEDURES PACKAGE';      GOTOXY(22,5);    DISPLAY1;
        WRITE(CHR(14));
        LINE:='(PSPP)';                  GOTOXY(34,7);    DISPLAY1;
        LINE:='FOR';                     GOTOXY(37,10);   DISPLAY1;
        LINE:='MICROCOMPUTER';           GOTOXY(27,12);   DISPLAY2;
        LINE:='Programmed by';           GOTOXY(33,15);   DISPLAY3;
        LINE:='David P. Kunkel';         GOTOXY(32,17);   DISPLAY3;

    END;  (* End of DRAW SCREEN *)

(*****************************************************************)

  PROCEDURE OVERVIEW;

(*****************************************************************)
(*                Procedures internal to OVERVIEW                *)
(*****************************************************************)

    PROCEDURE PAGE1;

        BEGIN
            GOTOXY(0,5);
            WRITELN('This package does Multivariate ',
                    'Statistical analysis.',CHR(13));
            WRITELN('The following modules are available:',
                     CHR(13));
            WRITELN('     1 - Data file preparation');
            WRITELN('         a - Create a new data file');
            WRITELN('         b - Save a file to disk');
            WRITELN('         c - Load a file from disk');
            WRITELN('         d - Modify data in a file');
            WRITELN('         e - Echo-check data in a file');
            WRITELN('     2 - Canonical Correlation analysis');
            WRITELN('     3 - Factor analysis');
        END;  (* End of PAGE 1 *)

(*****************************************************************)

    PROCEDURE PAGE2;

        BEGIN
            GOTOXY(0,3);
            WRITELN('Data file specifications are NOT under ',
                    'user control.',CHR(13));
            WRITELN('The following criteria exist:');
            WRITELN('    1 - Upper limit of ',MAXSIZE,
                    ' fields per record');
            WRITELN('    2 - Upper limit of ',MAXREC,
                    ' records in file');
            WRITELN('    3 - All records ',CHR(15),'MUST',
                   CHR(14),' be numeric',CHR(13),CHR(13));
            WRITELN('There are two types of user inputs:',
                     CHR(13));
            WRITELN('    1 - When asked to ''ENTER'' a value, ',
                    'you should type your');
            WRITELN('        response, then press the ',CHR(15),
                    'RETURN',CHR(14),' key.');
```

```
                WRITELN('    2 - When asked to ''PICK'' an option ',
                        'or asked a Yes or No question,');
                WRITELN('         the ',CHR(15),'RETURN',CHR(14),
                        ' key need not be pressed.',CHR(13));
                WRITELN(CHR(15),'NOTE:',CHR(14),' You will be asked',
                        ' if you have a printer on-line.  If you');
                WRITELN('       say YES and there is not, the ',
                        'program will run slower'');
            END;  (* End of PAGE 2 *)

(*****************************************************************)
(*                    Main body of OVERVIEW                      *)
(*****************************************************************)

        BEGIN
            WRITELN(CHR(12),' ':16,CHR(15),' PASCAL STATISTICAL ',
                    'PROCEDURES PACKAGE ',CHR(14));

            PAGE1;

            GOTOXY(22,22);
            WRITE('Press any key to continue    ');
            GETOPTION(OPT);
            ERASE(5,18);

            PAGE2;

            GOTOXY(22,22);
            WRITE('Press any key to continue    ');
            GETOPTION(OPT);

        END;  (* End of OVERVIEW *)

(*****************************************************************)

    PROCEDURE GETSPEC;

        BEGIN
            WRITELN(CHR(12),' ':16,CHR(15),' PASCAL STATISTICAL ',
                    'PROCEDURES PACKAGE ',CHR(14));
            GOTOXY(0,10);
            WRITE('Do you have a printer? ',CHR(15),' (Y/N)',
                    CHR(14));
            GETOPTION(OPT);
            WHILE (OPT<>'Y') AND (OPT<>'y') AND
                  (OPT<>'N') AND (OPT<>'n') DO
                      GETOPTION(OPT);

            IF (OPT='Y') OR (OPT='y') THEN
                PRINTER:=TRUE
            ELSE
                PRINTER:=FALSE;

        END;  (* End of GET printer SPECification *)

(*****************************************************************)
(*                    Main body of START UP                      *)
(*****************************************************************)

    BEGIN
    (*$R MAIN_UNIT *)                          (* Retain UNIT in memory *)

        DRAWSCREEN;

        GOTOXY(20,23);
        WRITE('Do you want introductory remarks? ',CHR(15),
              ' (Y/N)',CHR(14));
        GETOPTION(OPT);
        WHILE (OPT<>'Y') AND (OPT<>'y') AND
              (OPT<>'N') AND (OPT<>'n') DO
                  GETOPTION(OPT);

        IF (OPT='Y') OR (OPT='y') THEN
            OVERVIEW;

        GETSPEC;                               (* Ask if printer online *)

    END;  (* End of START UP *)

(***********************************************************************)
(*                      Initialization part of UNIT                    *)
(***********************************************************************)

END.
```

```
(*$S+*)

UNIT DATA_MOD;

INTERFACE
    USES TRANSCEND, MAIN_UNIT, MU_A, MU_B, MU_C, MU_D, MU_E;

    PROCEDURE DATAMODULE(VAR DATA:RAWDATA:VAR SPECS1:HEADER1;
                         VAR SPECS2:HEADER2;PRINTER:BOOLEAN);

IMPLEMENTATION

(*********************************************************************)
(*                     Main body of DATA_MODule                      *)
(*********************************************************************)

PROCEDURE DATAMODULE;

(*********************************************************************)
(*                                                                   *)
(*      This procedure handles data input, modification, and         *)
(*           data storage to disks previously formatted by           *)
(*           the PASCAL operating system.                            *)
(*                                                                   *)
(*      This procedure needs as input:                               *)
(*                                                                   *)
(*           DATA - Array of and for data storage                    *)
(*           SPECS1 - Array of field or variable names               *)
(*           SPECS2 - Array of field widths & file specs             *)
(*           PRINTER - Indicator of printer presence                 *)
(*                                                                   *)
(*      This procedure provides as output the above arrays           *)
(*           stored on disk or printed to screen and printer.        *)
(*                                                                   *)
(*********************************************************************)

    VAR
        OPT:                                (* Menu option           *)
              CHAR;
        DONE:                               (* Completion indicator  *)
              BOOLEAN;

(*********************************************************************)
(*                        Internal Procedure                         *)
(*********************************************************************)

    PROCEDURE GOTOPMENU;

        BEGIN
            WRITELN(CHR(12),' ':28,CHR(15),' DATA MODULE ',CHR(14));
            GOTOXY(0,5);
            WRITELN('Select desired option:');
            WRITELN('        1 - Create a new data file');
            WRITELN('        2 - Save a file to disk');
            WRITELN('        3 - Load a file from disk');
            WRITELN('        4 - Modify data in file');
            WRITELN('        5 - Echo-check data in file');
            WRITELN('        6 - Exit DATA MODULE');

            GETOPTION(OPT);
            WHILE (OPT<'1') OR (OPT>'6') DO
                 GETOPTION(OPT);

            CASE (OPT) OF
                '1': MAKEFILE(DATA,SPECS1,SPECS2);
                '2': SAVEFILE(DATA,SPECS1,SPECS2);
                '3': LOADDATA(DATA,SPECS1,SPECS2);
                '4': MODIFILE(DATA,SPECS1,SPECS2);
                '5': ECHOFILE(DATA,SPECS1,SPECS2,PRINTER);
                '6': DONE:=TRUE;
            END;  (* End of CASE *)
       END;  (* End of GO TOP MENU *)

(********************************************************************)
(*                        Main body of DATA MODULE                   *)
(********************************************************************)

  BEGIN
       DONE:=FALSE;
       WHILE NOT(DONE) DO
            GOTOPMENU;
  END;  (* End of DATA MODULE *)

(********************************************************************)
(*                      Initialization part of UNIT                  *)
(********************************************************************)

END.
```

```
(*$S++)

UNIT CANCOR_MOD;

INTERFACE
    USES TRANSCEND, MAIN_UNIT, MU_E, MU_F, MU_G, MU_H, MU_J, MU_K;

    PROCEDURE CANCOR(VAR DATA:RAWDATA;VAR SPECS1:HEADER1;
                     VAR SPECS2:HEADER2;PRINTER:BOOLEAN);

IMPLEMENTATION

(*******************************************************************)
(*                    Main body of CANCOR_MODule                 *)
(*******************************************************************)

PROCEDURE CANCOR;

(*******************************************************************)
(*                                                                 *)
(*       This procedure allows for the division of the DATA        *)
(*            array into two sets of variables.  It then           *)
(*            derives a linear combination from each set           *)
(*            such that the correlation between the two            *)
(*            linear combinations is maximized.                    *)
(*                                                                 *)
(*       This procedure needs as input:                            *)
(*                                                                 *)
(*            DATA - Array of data to be analyzed                  *)
(*            SPECS1 - Array of field or variable names            *)
(*            SPECS2 - Array of field widths & file specs          *)
(*            PRINTER - Indicator of printer presence              *)
(*                                                                 *)
(*       This procedure produces and outputs the canonical         *)
(*            variates and the correlations between them.          *)
(*                                                                 *)
(*            NOTE:  The rawdata is standardized but not           *)
(*                   automatically saved to disk.                  *)
(*                                                                 *)
(*******************************************************************)

    VAR
        I,                           (* Iteration counter             *)
        P,                           (* Number of criterion variables *)
        K,                           (* Number of predictor variables *)
        NUMREC,                      (* Number of records in DATA     *)
        WIDTH:                       (* Number of fields in DATA      *)
                INTEGER;
        OPT:                         (* Menu option                   *)
                CHAR;
        GROUP:                       (* Variable type designations    *)
                HEADER2;
        XBAR,                        (* Variable means                *)
        SDEV,                        (* Variable standard deviations  *)
        EIGVAL,                      (* Calculated eigenvalues        *)
        CANCORS,                     (* Canonical correlations        *)
        WILKSL,                      (* Wilk's Lambda statistics      *)
        CHISQR:                      (* CHI Square statistics         *)
                VECTOR;
        CORRMAT,                     (* Sample Correlation Matrix     *)
        EIGVEC,                     (* Eigenvectors/Y Canonical wts  *)
        A,                          (* Matrix to get eigenvalues of  *)
        BETA:                       (* Scaled X Canonical weights    *)
              MATRIX;
        FEAS,                       (* Feasible matrix inversion     *)
        DONE:                       (* Completion indicator          *)
              BOOLEAN;

(*******************************************************************)
(*                         Internal Procedures                      *)
(*******************************************************************)

    PROCEDURE GETVARSNSTATS;

        BEGIN
        (*$R MU_F *)                          (* Retain UNIT in memory *)

            ASSIGNVARIABLES(SPECS1,SPECS2,GROUP,1,FEAS);

            IF (FEAS) THEN
                BEGIN
                    P:=GROUP[-1];
                    K:=GROUP[0];
                    ERASE(5,18);

                    IF (PRINTER) THEN
                        BEGIN
                            WRITE(PTR,CHR(12),CHR(18),CHR(14),
                                  ' ':7,'CANONICAL CORRELATION',
                                  CHR(20),CHR(15));
                            FOR I:=1 TO 3 DO
                                 WRITELN(PTR);
                        END;

                    GOTOXY(0,22);
                    WRITELN('Calculating MEANS & STANDARD ',
                            'DEVIATIONS. . .Please stand by   ');

                    CALCULATE(XBAR,SDEV,DATA,SPECS1,GROUP,NUMREC,
                              WIDTH,PRINTER);

                    ERASE(22,1);
                    GOTOXY(0,20);
                    WRITELN('Select desired option:');
                    WRITELN('       1 - Proceed with ',
                                 'standardization');
                    WRITELN('       2 - Exit CANCOR routine');
                    GETOPTION(OPT);
                    WHILE (OPT<>'1') AND (OPT<>'2') DO
                        GETOPTION(OPT);
                    ERASE(20,3);
                    IF (OPT='2') THEN
                        DONE:=TRUE;
                END
            ELSE
                DONE:=TRUE;

        END;  (* End of GET VARiableS aNd STATisticS *)

(*******************************************************************)
```

```
    PROCEDURE STANDNGETCORRMAT;

        BEGIN
        (*$R MU_G *)                          (* Retain UNIT in memory *)

            ERASE(5,18);
            GOTOXY(0,20);
            WRITELN('Standardizing designated variables. . .',
                    'Please stand by    ');

            STANDARDIZE(DATA,XBAR,SDEV,GROUP,NUMREC,WIDTH,'2');

            GOTOXY(0,22);
            WRITELN(CHR(7),'Generating Correlation Matrix. . .',
                           'Please stand by    ');

            GENMATRIX(DATA,CORRMAT,SPECS1,GROUP,NUMREC,
                      WIDTH,PRINTER);

            GOTOXY(0,20);
            WRITELN('Select desired option:');
            WRITELN('         1 - Proceed with statistics ',
                          'calculation');
            WRITELN('         2 - Exit CANCOR routine');
            GETOPTION(OPT);
            WHILE (OPT<>'1') AND (OPT<>'2') DO
                GETOPTION(OPT);

            IF (OPT='2') THEN
                DONE:=TRUE;

        END;  (* End of STANDardize aNd GET CORRelation MATrix *)

(*********************************************************************)

    PROCEDURE CALCULATESTATS;

        BEGIN
        (*$R MU_H *)                          (* Retain UNIT in memory *)

            ERASE(2,21);
            GOTOXY(0,22);
            WRITELN('Calculating Eigenvalues. . .',
                    'Please stand by    ');

            PREPTOEIG(CORRMAT,P,K,A,FEAS);

            IF NOT(FEAS) THEN         (* Multi-collinearity trap    *)
                BEGIN
                    GOTOXY(1,22);
                    WRITELN(CHR(15),'WARNING:',CHR(14),'  Data is ',
                            'multicollinear. CANCOR can not proceed');
                    WRITE(' ':11,'Press any key to exit    ');
                    GOTOXY(0,22);
                    GETOPTION(OPT);
                    DONE:=TRUE;
                END;

            IF NOT(DONE) THEN         (* Continue calculating stats *)
```

```
                    BEGIN
                        EIGEN(P,A,EIGVEC,EIGVAL);

                        ERASE(22,1);
                        GOTOXY(0,20);
                        WRITELN(CHR(7),'Calculating Canonical ',
                               'Correlations, Wilk''s Lambda, and ',
                               'CHI Square. . .',CHR(13));

                        GETCANCORSTATS(EIGVAL,CANCORS,WILKSL,CHISQR,
                                       NUMREC,P,K,PRINTER);

                        GOTOXY(0,20);
                        WRITELN('Select desired option:');
                        WRITELN('        1 - Proceed with Canonical ',
                                'Variable calculation');
                        WRITELN('        2 - Exit CANCOR routine ');
                        GETOPTION(OPT);
                        WHILE (OPT<>'1') AND (OPT<>'2') DO
                            GETOPTION(OPT);

                        IF (OPT='2') THEN
                            DONE:=TRUE;
                    END;

            END;  (* End of CALCULATE STATisticS *)

    (*********************************************************************)

        PROCEDURE GETSTRUCTCORR;

            BEGIN
                ERASE(5,18);
                GOTOXY(0,22);
                WRITE('Calculating Canonical Variate Coefficients',
                      '. . .Please stand by   ');

                GETCVCS(CANCORS,EIGVEC,BETA,CORRMAT,SPECS1,
                        GROUP,PRINTER);

                GOTOXY(0,22);
                WRITE('Calculating Canonical Variate Scores. . .',
                      'Please stand by   ');

                GETCVSS(DATA,GROUP,EIGVEC,BETA,NUMREC,
                        WIDTH,PRINTER);

                GOTOXY(0,22);
                WRITE('Calculating Structure Correlations. . .',
                      'Please stand by   ');

                STRUCTURECORR(EIGVEC,BETA,CORRMAT,EIGVAL,SPECS1,
                              GROUP,WIDTH,PRINTER);

            END;  (* End of GET STRUCTure CORRelations *)

    (*********************************************************************)
    (*                      Main body of CANCOR                          *)
    (*********************************************************************)
```

```
BEGIN
    NUMREC:=SPECS2[-1];          (* Initialize parameters          *)
    WIDTH:=SPECS2[0];
    DONE:=FALSE;
    FEAS:=TRUE;

    WRITELN(CHR(12),' ':20,CHR(15),' CANONICAL CORRELATION ',
           'ROUTINE ',CHR(14));
    GOTOXY(0,20);
    WRITELN('Select desired option:');
    WRITELN('        1 - Proceed with variable selection');
    WRITELN('        2 - Exit CANCOR routine');
    GETOPTION(OPT);
    WHILE (OPT<>'1') AND (OPT<>'2') DO
        GETOPTION(OPT);
    IF (OPT='2') THEN
        DONE:=TRUE;
    ERASE(20,3);

    IF NOT(DONE) THEN            (* Input & Calculate statistics *)
        GETVARSNSTATS;

    IF NOT(DONE) THEN            (* Standardize & Get Corr Mat    *)
        STANDNGETCORRMAT;

    IF NOT(DONE) THEN            (* Calculate statistics          *)
        CALCULATESTATS;

    IF NOT(DONE) THEN            (* Get Structure Correlations    *)
        GETSTRUCTCORR;
END;  (* End of CANCOR *)

(**********************************************************************)
(*                   Initialization part of UNIT                      *)
(**********************************************************************)

END.
```

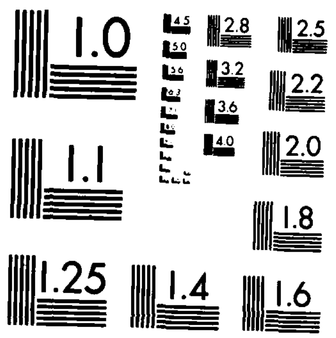
```
(*$S+*)

UNIT FACTOR_MOD;

INTERFACE
    USES TRANSCEND, APPLESTUFF, MAIN_UNIT, MU_E, MU_F, MU_G,
                                          MU_H, MU_I, MU_K;

    PROCEDURE FACTOR(VAR DATA:RAWDATA;VAR SPECS1:HEADER1;
                     VAR SPECS2:HEADER2;PRINTER:BOOLEAN);

IMPLEMENTATION

(****************************************************************)
(*                  Main part of FACTOR_MODule                 *)
(****************************************************************)

PROCEDURE FACTOR;

(****************************************************************)
(*                                                              *)
(*      This procedure looks for an underlying pattern of       *)
(*           relationships between members of a designated      *)
(*           set of variables so that a possible reduction      *)
(*           to a smaller set of factors or components can      *)
(*           be done.                                           *)
(*                                                              *)
(*      This procedure needs as input:                          *)
(*           DATA - Array of data to be analyzed                *)
(*           SPECS1 - Array of field or variable names          *)
(*           SPECS2 - Array of field widths & file specs        *)
(*           PRINTER - Indicator of printer presence            *)
(*                                                              *)
(*      This procedure produces and output the Factor           *)
(*           Loadings, Communalities, Coefficients, and         *)
(*           Scores for a given set of data.                    *)
(*                                                              *)
(****************************************************************)

    VAR
        I,                          (* Iteration counter            *)
        N,                          (* Number of Factors            *)
        NS,                         (* Number of Significant Factors *)
        NUMREC,                     (* Number of data records       *)
        WIDTH:                      (* Number of record fields      *)
                 INTEGER;
        OPT:                        (* Menu option                  *)
                 CHAR;
        GROUP:                      (* Identifies designated set    *)
                 HEADER2;
        FEAS,                       (* Feasibility indicator        *)
        DONE:                       (* Completion indicator         *)
                 BOOLEAN;
        XBAR,                       (* Array of field means         *)
        SDEV,                       (* Array of field Standard Dev. *)
        EIGVAL:                     (* Array of Eigenvalues         *)
                 VECTOR;
        CORRMAT,                    (* Correlation matrix           *)
        EIGVEC,                     (* Array of Eigenvectors        *)
        FACTCOEF:                   (* Array of Factor Coefficients *)
```

```
          MATRIX;

(*****************************************************************)
(*                       Internal Procedures                     *)
(*****************************************************************)

   PROCEDURE INPUTNCALCSTATS;

      BEGIN
      (*$R MU_F *)                         (* Retain UNIT in memory *)

         ASSIGNVARIABLES(SPECS1,SPECS2,GROUP,2,FEAS);

         IF (FEAS) THEN
            BEGIN
               IF (PRINTER) THEN
                  BEGIN
                     WRITE(PTR,CHR(12),CHR(18),CHR(14),' ':10,
                           'FACTOR ANALYSIS',CHR(20),CHR(15));
                     FOR I:=1 TO 3 DO
                        WRITELN(PTR);
                  END;

               N:=GROUP[0];
               ERASE(5,18);
               GOTOXY(0,22);
               WRITE('Calculating Means & Standard Deviations',
                     '. . .Please stand by   ');

               CALCULATE(XBAR,SDEV,DATA,SPECS1,GROUP,NUMREC,
                         WIDTH,PRINTER);

               GOTOXY(0,20);
               WRITELN('Select desired option:');
               WRITELN('        1 - Proceed with ',
                           'Standardization');
               WRITELN('        2 - Exit FACTOR routine',' ':30);
               GETOPTION(OPT);
               WHILE (OPT<>'1') AND (OPT<>'2') DO
                  GETOPTION(OPT);
               IF (OPT='2') THEN
                  DONE:=TRUE;
            END
         ELSE
            DONE:=TRUE;
      END;  (* End of INPUT variables & CALCulate STATisticS *)

(*****************************************************************)

   PROCEDURE STANDNGETCORRMAT;

      BEGIN
      (*$R MU_G *)                         (* Retain UNIT in memory *)

         ERASE(5,18);
         GOTOXY(0,20);
         WRITE('Standardizing data. . .Please stand by   ');

         STANDARDIZE(DATA,XBAR,SDEV,GROUP,NUMREC,WIDTH,'2');
```

```
          GOTOXY(0,22);
          WRITE(CHR(7),'Generating Correlation Matrix. . .',
                   'Please stand by   ');

          GROUP[-1]:=TRUNC(N/2.0);           (* Partitions        *)
          GROUP[0]:=N-GROUP[-1];

          GENMATRIX(DATA,CORRMAT,SPECS1,GROUP,NUMREC,
                    WIDTH,PRINTER);

          GROUP[0]:=N;
          GOTOXY(0,20);
          WRITELN('Select desired option:');
          WRITELN('       1 - Proceed with FACTOR calculation');
          WRITELN('       2 - Exit FACTOR routine');
          GETOPTION(OPT);
          WHILE (OPT<>'1') AND (OPT<>'2') DO
              GETOPTION(OPT);

          IF (OPT='2') THEN
              DONE:=TRUE;

      END;  (* End of STANDardize & GET CORRelation MATrix *)

(*******************************************************************)

    PROCEDURE GETSTATSNSCORES;

      BEGIN
          ERASE(2,21);
          GOTOXY(0,22);
          WRITE('Calculating Eigenvalues. . .',
                'Please stand by   ');

          EIGEN(N,CORRMAT,EIGVEC,EIGVAL);

          ERASE(22,1);

          SELECTFACTORS(NS,NUMREC,EIGVAL,GROUP,PRINTER);

          GOTOXY(0,22);
          WRITE('Calculating Factor Loadings. . .',
                'Please stand by   ');

          FACTORMAT(EIGVAL,EIGVEC,FACTCOEF,SPECS1,GROUP,
                    WIDTH,PRINTER);

          GOTOXY(0,22);
          WRITE('Calculating Factor Scores. . .',
                'Please stand by   ');

          GETFACTSCORES(DATA,FACTCOEF,GROUP,NUMREC,
                        WIDTH,PRINTER);

      END;  (* End of GET STATistics & factor SCORES *)

(*******************************************************************)
(*                     Main body of FACTOR routine                  *)
(*******************************************************************)
```

AD-A141 049 PASCAL STATISTICAL PROCEDURES PACKAGE (PSPP)(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGINEERING D P KUNKEL DEC 83 AFIT/GSO/OS/83D-4 2/3

UNCLASSIFIED F/G 9/2 NL



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

```
    BEGIN
        NUMREC:=SPECS2[-1];
        WIDTH:=SPECS2[0];
        DONE:=FALSE;
        FEAS:=TRUE;

        WRITELN(CHR(12),' ':28,CHR(15),' FACTOR ANALYSIS ROUTINE ',
                CHR(14));
        GOTOXY(0,20);
        WRITELN('Select desired option:');
        WRITELN('       1 - Proceed with variable selection');
        WRITELN('       2 - Exit FACTOR routine');
        GETOPTION(OPT);
        WHILE (OPT<>'1') AND (OPT<>'2') DO
            GETOPTION(OPT);
        IF (OPT='2') THEN
            DONE:=TRUE;
        ERASE(20,3);

        IF NOT(DONE) THEN
            INPUTNCALCSTATS;

        IF NOT(DONE) THEN
            STANDNGETCORRMAT;

        IF NOT(DONE) THEN
            GETSTATSNSCORES;

    END;  (* End of FACTOR routine *)

(*****************************************************************)
(*                  Initialization part of UNIT                  *)
(*****************************************************************)

END.
```

```
(*$S+*)

UNIT MAIN_UNIT;  INTRINSIC CODE 27 DATA 28;

INTERFACE

    CONST
        MAXREC  = 200;        (* Maximum number of records per file  *)
        MAXSIZE =  10;        (* Maximum number of fields per record *)

    TYPE
        VECTOR  = ARRAY[1..MAXSIZE] OF REAL;
        MATRIX  = ARRAY[1..MAXSIZE] OF VECTOR;
        HEADER1 = ARRAY[0..MAXSIZE] OF STRING[15];
        HEADER2 = ARRAY[-1..MAXSIZE] OF INTEGER;
        RAWDATA = ARRAY[1..MAXREC,1..MAXSIZE] OF REAL;

    VAR
        PRINTER:              (* Flag set indicates printer presence *)
                  BOOLEAN;
        DATAFILE,             (* Used for data transfer to/from disk *)
        PTR:                  (* File to hold text to be printed     *)
                  TEXT;
        SPECS1:               (* Array of field and variable names   *)
                  HEADER1;
        SPECS2:               (* Array of field widths & file specs  *)
                  HEADER2;
        DATA,                 (* Array of data used by all routines  *)
        SCORES:               (* Array of FACTOR or CANCOR scores    *)
                  RAWDATA;

        PROCEDURE GETOPTION(VAR OPT:CHAR);

        PROCEDURE ERASE(ROW,LINES:INTEGER);

IMPLEMENTATION

(*****************************************************************)
(*                     Main body of MAIN_UNIT                    *)
(*****************************************************************)

PROCEDURE GETOPTION;

(*****************************************************************)
(*                                                               *)
(*      This procedure rings a bell to alert the user to         *)
(*           a required input, then accepts 1 character.         *)
(*           It is used after menu displays and as means         *)
(*           of delaying operation until signaled by user.       *)
(*                                                               *)
(*****************************************************************)

    BEGIN
        WRITE(CHR(7));                (* Ring bell to alert user    *)
        READ(KEYBOARD,OPT);           (* Accept a single character  *)
    END;  (* End of GETOPTION *)

(*****************************************************************)

PROCEDURE ERASE;
```

```
(*****************************************************************)
(*                                                               *)
(*        This procedure erases LINES from the screen,           *)
(*             starting from screen position (0,ROW).            *)
(*                                                               *)
(*****************************************************************)

    VAR I:  INTEGER;                          (* Iteration counter  *)

    BEGIN
         GOTOXY(0,ROW);
         FOR I:=1 TO LINES DO
              WRITELN(CHR(29));                (* Erases one line    *)
    END;  (* End of ERASE *)

(*****************************************************************)
(*                Initialization part of UNIT                    *)
(*****************************************************************)

END.
```

```
(*$S+*)

UNIT MU_A; INTRINSIC CODE 11;

INTERFACE
    USES MAIN_UNIT;

    PROCEDURE MAKEFILE(VAR DATA:RAWDATA;VAR SPECS1:HEADER1;
                       VAR SPECS2:HEADER2);

IMPLEMENTATION

(*****************************************************************)
(*                      Main body of MU_A                        *)
(*****************************************************************)

PROCEDURE MAKEFILE;

(*****************************************************************)
(*                                                               *)
(*        This procedure accepts as input the contents of a      *)
(*             data array, specifications as to the size and     *)
(*             width of the data fields and names for each field. *)
(*                                                               *)
(*        This procedure returns as output:                      *)
(*                                                               *)
(*             DATA - Array of raw data as input by user         *)
(*             SPECS1 - Array of field or variable names         *)
(*             SPECS2 - Array of field widths                    *)
(*                                                               *)
(*****************************************************************)

    VAR
        I,                                  (* Iteration counter *)
        COLS,                               (* Number of columns *)
        FIELD,                              (* Field identifier  *)
        INDEX,                              (* Index into arrays *)
        NUMREC,                             (* Number of records *)
        ROW,                                (* Row on screen     *)
        WIDTH:                              (* Number of fields  *)
            INTEGER;
        VALUE:                              (* User inputed data *)
            REAL;
        DONE:                               (* Exit  indicator   *)
           BOOLEAN;
        OPT:                                (* Menu option       *)
           CHAR;

(*****************************************************************)
(*                      Internal Procedures                      *)
(*****************************************************************)

    PROCEDURE PAGE1;

        BEGIN
            WRITELN('Before entering the data, modify it as ',
                    'follows: ',CHR(13));
            WRITELN('        1 - All entries must be numeric.  ',
                    'See user''s manual for help');
            WRITELN('             on converting letters to numbers.');
            WRITELN;
            WRITELN('        2 - Upper limit of ',MAXSIZE,' fields ',
                    'or variables per record.');
            WRITELN;
            WRITELN('        3 - Upper limit of ',MAXREC,' records ',
                    'per data file.');
            WRITELN;
            WRITELN('        4 - Upper limit of 80 characters per ',
                    'record.  This includes all');
            WRITELN('            decimal points and spaces between ',
                    'fields.');
            WRITELN;
            WRITELN('        5 - The first field of any record can ',
                    'not be 9999.  This value');
            WRITELN('            is used to signify data entry ',
                    'completion.');
        END;  (* End of PAGE 1 *)

(****************************************************************)

    PROCEDURE PAGE2;

        BEGIN
            WRITELN('Order of entry is as follows:',CHR(13),CHR(13));
            WRITELN('  First - The number of fields or variables ',
                    'is requested.',CHR(13));
            WRITELN('   Next - The name and width of each field is ',
                    'requested.  Remember to');
            WRITELN('          leave room for the largest value in ',
                    'each field.  Also, the');
            WRITELN('          field name should be less than or ',
                    'equal to the width, or it');
            WRITELN('          will be truncated to fit.',CHR(13));
            WRITELN('Finally - Each record is entered, one field at',
                    ' a time.  After the last');
            WRITELN('          field is entered, you will be asked ',
                    'if any changes need to be');
            WRITELN('          made.  Enter 9999 in the first field',
                    ' to signify completion.');
        END;  (* End of PAGE 2 *)

(****************************************************************)

  PROCEDURE SHOWINSTRUCTIONS;

      BEGIN
          WRITELN(CHR(12),' ':23,CHR(15),' DATA ENTRY ',
                  'INSTRUCTIONS ',CHR(14));
          GOTOXY(0,5);

          PAGE1;

          GOTOXY(22,22);
          WRITE('Press any key to continue   ');
          GETOPTION(OPT);
          ERASE(5,18);
          GOTOXY(0,5);

          PAGE2;
```

```
            GOTOXY(22,22);
            WRITE('Press any key to continue   ');
            GETOPTION(OPT);
        END;  (* End of SHOW INSTRUCTIONS *)

(***************************************************************)

(*$I PSPP:GATHERDATA *)            (* Include file in compilation *)

(***************************************************************)

    PROCEDURE DISPLAYMENU;

        BEGIN
            WRITELN(CHR(12),' ':25,CHR(15),' DATA ENTRY PROCEDURE ',
                    CHR(14));
            GOTOXY(0,5);
            WRITELN('Select desired option:');
            WRITELN('         1 - Display instructions');
            WRITELN('         2 - Enter raw data');
            WRITELN('         3 - Exit DATA ENTRY procedure');

            GETOPTION(OPT);
            WHILE (OPT<'1') OR (OPT>'3') DO
                GETOPTION(OPT);

            CASE (OPT) OF
              '1':SHOWINSTRUCTIONS;
              '2':GATHERDATA;
              '3':DONE:=TRUE;
            END;

        END;  (* End of DISPLAY MENU *)

(***************************************************************)
(*                    Main body of MAKEFILE                    *)
(***************************************************************)

    BEGIN
        DONE:=FALSE;
        WHILE NOT(DONE) DO
            DISPLAYMENU;
    END;  (* End of MAKE FILE *)

(***************************************************************)
(*                 Initialization part of UNIT                 *)
(***************************************************************)

END.
```

```
PROCEDURE GATHERDATA;

(**********************************************************************)
(*                                                                    *)
(*        This procedure is the main working section used            *)
(*             by MAKEFILE to structure and fill the data            *)
(*             and specification arrays.                              *)
(*                                                                    *)
(**********************************************************************)

(**********************************************************************)
(*                  Procedures internal to GATHERDATA                 *)
(**********************************************************************)

PROCEDURE DISPLAYSPECS;

    BEGIN
        GOTOXY(0,7);
        FOR I:=1 TO WIDTH DO
            WRITELN(I:7,SPECS2[I]:12,' ':7,SPECS1[I],CHR(29));
    END;  (* End of DISPLAY SPECS *)

(**********************************************************************)

PROCEDURE HANDLEINVALID;

    BEGIN
    (*$I-*)
        WRITE(CHR(8),' ':20);
        GOTOXY(1,20);
        WRITE(CHR(15),'WARNING:',CHR(14),'  Value must ',
             'be a number. Press any key to continue   ');
        GOTOXY(0,20);
        GETOPTION(OPT);
        ERASE(20,1);
        GOTOXY(33,ROW);
        RESET(INPUT);
        READ(VALUE);
    (*$I+*)
    END;  (* End of HANDLE INVALID entry *)

(**********************************************************************)

PROCEDURE ENTERVALUE;

    VAR NAME: STRING;                         (* Field name to enter   *)

    BEGIN
    (*$I-*)
        ROW:=FIELD+6;
        GOTOXY(0,ROW);
        NAME:=SPECS1[FIELD];
        IF (LENGTH(NAME)>15) THEN             (* Truncate names to fit *)
            NAME:=COPY(NAME,1,15);
        WRITELN(FIELD:4,NAME:16,SPECS2[FIELD]:7);
        GOTOXY(33,ROW);
        RESET(INPUT);
        READ(VALUE);
        WHILE (IORESULT=14) DO
```

```
          HANDLEINVALID;
       DATA[NUMREC,FIELD]:=VALUE;
    (*$I+*)
    END;  (* End of ENTER VALUE *)

(*****************************************************************)

PROCEDURE CHANGEVALUE;

    VAR FIELD: INTEGER;                    (* Field to be changed    *)

    BEGIN
    (*$I-*)
        GOTOXY(0,19);
        WRITELN('Enter field to change:');
        WRITELN('  (0 = Skip change)');
        RESET(INPUT);
        READ(FIELD);

        WHILE (IORESULT=14) OR (FIELD<0) OR (FIELD>WIDTH) DO
            BEGIN
                GOTOXY(0,22);
                WRITE(CHR(7),'Bad field number.  Press any ',
                       'key to try again   ');
                GETOPTION(OPT);
                ERASE(21,2);
                GOTOXY(0,21);
                RESET(INPUT);
                READ(FIELD);
            END; (* End of Invalid Index *)

        ERASE(19,4);
        IF (FIELD<>0) THEN                         (* Make change *)
            BEGIN
                ROW:=FIELD+6;
                GOTOXY(56,ROW);
                WRITE(CHR(15),'<= Enter new value:',CHR(14));
                GOTOXY(33,ROW);
                RESET(INPUT);
                READ(VALUE);
                WHILE (IORESULT=14) DO
                    HANDLEINVALID;
                DATA[INDEX,FIELD]:=VALUE;
                GOTOXY(56,ROW);
                WRITELN(CHR(29));
            END;
    (*$I+*)
    END;  (* End of CHANGE VALUE *)

(*****************************************************************)

PROCEDURE GETFLDWIDTH;

    BEGIN
    (*$I-*)
        GOTOXY(60,5);                      (* Display column status *)
        WRITE(CHR(15),COLS:3,CHR(14));
        GOTOXY(76,5);
        WRITE(CHR(15),(80-COLS):3,CHR(14));
```

```
          GOTOXY(18,ROW);
          RESET(INPUT);
          READ(SPECS2[INDEX]);

          WHILE (IORESULT=14) OR (SPECS2[INDEX]<8) OR
                   (SPECS2[INDEX]>15) OR (COLS+SPECS2[INDEX]>80) DO
              BEGIN
                  GOTOXY(16,ROW);
                  WRITELN(CHR(29));

                  GOTOXY(56,ROW);                      (* Error Messages *)
                  IF (SPECS2[INDEX]<8) THEN
                      WRITE(CHR(15),'Must be at least 8',CHR(14));

                  IF (SPECS2[INDEX]>15) OR (COLS+SPECS2[INDEX]>80) THEN
                      BEGIN
                          IF (SPECS2[INDEX]>15) AND
                                          (COLS+SPECS2[INDEX]>80) THEN
                              BEGIN
                                  IF (80-COLS<15) THEN
                                      WRITE(CHR(15),'Must be no more ',
                                             'than ',80-COLS,CHR(14))
                                  ELSE
                                      WRITE(CHR(15),'Must be no more ',
                                             'than 15',CHR(14))
                              END
                          ELSE
                              BEGIN
                                  IF (SPECS2[INDEX]>15) THEN
                                      WRITE(CHR(15),'Must be no more ',
                                             'than 15',CHR(14))
                                  ELSE
                                      WRITE(CHR(15),'Must be no more ',
                                             'than ',80-COLS,CHR(14))
                              END
                      END; (* End of Error Messages *)

                  GOTOXY(18,ROW);
                  RESET(INPUT);
                  READ(SPECS2[INDEX]);
              END;  (* End of Bad Width *)

          COLS:=COLS+SPECS2[INDEX];
          GOTOXY(60,5);                       (* Display column status *)
          WRITE(CHR(15),COLS:3,CHR(14));
          GOTOXY(76,5);
          WRITE(CHR(15),(80-COLS):3,CHR(14));

      (*$I+*)
      END;  (* End of GET FIELD WIDTH *)

(*******************************************************************)

PROCEDURE GETFLDNAME;

      BEGIN
      (*$I-*)
          GOTOXY(26,ROW);
          WRITE(CHR(29));
          RESET(INPUT);
          READLN(SPECS1[INDEX]);

          IF (LENGTH(SPECS1[INDEX])>SPECS2[INDEX]) THEN
              WHILE (LENGTH(SPECS1[INDEX])>SPECS2[INDEX]) AND
                    (POS(' ',SPECS1[INDEX])=1) DO
                        DELETE(SPECS1[INDEX],1,1);

          IF (LENGTH(SPECS1[INDEX])>SPECS2[INDEX]) THEN
              SPECS1[INDEX]:=COPY(SPECS1[INDEX],1,SPECS2[INDEX]);
      (*$I+*)
      END;  (* End of GET FIELD NAME *)

(********************************************************************)

      PROCEDURE GETWIDTH;

          VAR
              OPT:                          (* Menu option              *)
                     CHAR;
              BAD:                          (* Invalid designator       *)
                     BOOLEAN;

(********************************************************************)
(*                        Internal Procedures                       *)
(********************************************************************)

          PROCEDURE BADWIDTH;

              BEGIN
                  WRITE(CHR(26));
                  GOTOXY(1,20);
                  WRITELN(CHR(15),'WARNING:',CHR(14),
                          '  You must enter an integer between ',
                          '1 and ',MAXSIZE,'.',CHR(13));
                  WRITE(' ':11,'Press any key to try again   ');
                  GOTOXY(0,20);
                  GETOPTION(OPT);
                  ERASE(20,3);
              END;

(********************************************************************)

          PROCEDURE GOODWIDTH;

               BEGIN
                   GOTOXY(0,9);
                   WRITELN('Do you want to stay with ',CHR(15),WIDTH,
                           CHR(14),' fields?',CHR(13));
                   WRITELN('Select desired option:');
                   WRITELN('       1 - Change size');
                   WRITELN('       2 - Go on to field definition');
                   GETOPTION(OPT);
                   WHILE (OPT<>'1') AND (OPT<>'2') DO
                       GETOPTION(OPT);

                   ERASE(7,7);
                   IF (OPT='2') THEN
                       BEGIN
                           SPECS2[0]:=WIDTH;
                           BAD:=FALSE;
                    END;
                END;

(*****************************************************************)
(*                      Main body of GETWIDTH                    *)
(*****************************************************************)

        BEGIN
        (*$I-*)
            BAD:=TRUE;

            WRITELN(CHR(12),'How many variables or fields ',
                            'of data do you have?');
            GOTOXY(0,5);
            WRITELN('Enter an integer of ',MAXSIZE,
                    ' or less:',CHR(7));

            WHILE (BAD) DO
                BEGIN
                    GOTOXY(0,7);
                    RESET(INPUT);
                    READ(WIDTH);

                    IF (IORESULT=14) OR (WIDTH<1) OR
                                            (WIDTH>MAXSIZE) THEN
                        BADWIDTH
                    ELSE
                        GOODWIDTH;

                END;  (* End of WHILE loop *)
        (*$I+*)
        END;  (* End of GET WIDTH *)

(*****************************************************************)

    PROCEDURE GETSPECS;

        VAR OPT: CHAR;                          (* Menu option      *)

(*****************************************************************)
(*                        Internal Procedures                    *)
(*****************************************************************)

        PROCEDURE INITIALENTRY;

            BEGIN
                WHILE (INDEX<WIDTH) AND (COLS<73) DO
                    BEGIN
                        INDEX:=INDEX+1;
                        ROW:=INDEX+6;
                        GOTOXY(0,ROW);
                        WRITE(INDEX:7);

                        GETFLDWIDTH;

                        GETFLDNAME;
                    END;

                IF (INDEX<WIDTH) THEN
                    BEGIN
                    WIDTH:=INDEX;
                    GOTOXY(1,20);
                    WRITELN(CHR(15),'WARNING:',CHR(14),'  You ',
                            'are limited to ',WIDTH,' variables.',
                            ' There isn''t room for more because');
                    WRITELN(' ':11,'you haven''t left room for ',
                            'more on the 80-column screen line.');
                    WRITE(' ':11,'Press any key to continue');
                    GOTOXY(0,20);
                    GETOPTION(OPT);
                    ERASE(20,3);
                END;
            END;

(*****************************************************************)

        PROCEDURE CHANGEDESIRED;

            BEGIN
            (*$I-*)
                ERASE(18,5);
                GOTOXY(0,20);
                WRITELN('Enter FIELD to change, new WIDTH, ',
                        'and new NAME (0 = No Change)');
                ROW:=18;
                GOTOXY(6,ROW);
                RESET(INPUT);
                READ(INDEX);

                WHILE (IORESULT=14) OR (INDEX<0) OR (INDEX>WIDTH) DO
                    BEGIN
                        GOTOXY(50,ROW);
                        WRITE(CHR(7),CHR(15),'WARNING:',CHR(14),
                              '  Must be from 0 to',WIDTH:3);
                        GOTOXY(60,ROW+1);
                        WRITE('Press any key');
                        GOTOXY(49,ROW);
                        GETOPTION(OPT);

                        GOTOXY(6,ROW);             (* Erase message *)
                        WRITE(CHR(29));
                        GOTOXY(60,ROW+1);
                        WRITE(CHR(29));

                        GOTOXY(6,ROW);
                        RESET(INPUT);
                        READ(INDEX);
                    END;  (* End of Bad Index *)

                IF (INDEX<>0) THEN                 (* Change Field *)
                    BEGIN
                        COLS:=COLS-SPECS2[INDEX];

                        GETFLDWIDTH;

                        GETFLDNAME;

                        ERASE(18,5);
                        DISPLAYSPECS;
                    END;
            IF (COLS<73) THEN                    (* Allow addition *)
                BEGIN
                    GOTOXY(0,19);
                    WRITELN('Select desired option:');
                    WRITELN('        1 - Add a variable');
                    WRITELN('        2 - Stay with ',WIDTH,
                                ' variables');
                    GETOPTION(OPT);
                    IF (OPT<>'1') AND (OPT<>'2') THEN
                        GETOPTION(OPT);
                    ERASE(19,3);

                    IF (OPT='1') THEN
                        BEGIN
                            GOTOXY(50,5);
                            WRITE(CHR(15),'COLS USED:',COLS:3,
                                  '   COLS LEFT:',(80-COLS):3,
                                  CHR(14));

                            INDEX:=WIDTH;
                            WIDTH:=WIDTH+1;

                            INITIALENTRY;
                        END;
                    ERASE(7,16);
                    DISPLAYSPECS;
                END;

            GOTOXY(0,19);
            WRITELN('Select desired option:');
            WRITELN('        1 - Change a field');
            WRITELN('        2 - Go on to data entry');
            GETOPTION(OPT);
            WHILE (OPT<>'1') AND (OPT<>'2') DO
                GETOPTION(OPT);
        END;

(*****************************************************************)
(*                     Main body of GETSPECS                     *)
(*****************************************************************)

    BEGIN
        WIDTH:=SPECS2[0];            (* Initialize parameters   *)
        COLS:=0;

        WRITELN(CHR(12),'Now enter widths and names for ',WIDTH,
                ' fields.  The widths ',CHR(15),'must',CHR(14),
                ' be at least 8');
        WRITELN('and no more than 15.  This includes room for ',
                '6 significant digits.  Names');
        WRITELN('should be no wider than their field.  ',
                'Finally, remember the upper limit');
        WRITELN('of 80 characters per record, in order to ',
                'display on one screen line.');

        GOTOXY(0,5);
        WRITE(CHR(15),'FIELD NUMBER','WIDTH':9,
                'NAME':8,CHR(14));
```

```
          GOTOXY(50,5);
          WRITELN(CHR(15),'COLS USED:',COLS:3,'   COLS LEFT:',
                 (80-COLS):3,CHR(14));

          INDEX:=0;
          INITIALENTRY;

          WRITELN(CHR(12),'Current field specifications:');
          GOTOXY(0,5);
          WRITE(CHR(15),'FIELD NUMBER','WIDTH':9,
                 'NAME':8,CHR(14));
          GOTOXY(50,5);
          WRITELN(CHR(15),'COLS USED:',COLS:3,'   COLS LEFT:',
                 (80-COLS):3,CHR(14));

          DISPLAYSPECS;

          GOTOXY(0,19);
          WRITELN('Select desired option:');
          WRITELN('       1 - Change or add a field');
          WRITELN('       2 - Go on to data entry
          GETOPTION(OPT);
          WHILE (OPT<>'1') AND (OPT<>'2') DO
              GETOPTION(OPT);

          WHILE (OPT='1') DO
              CHANGEDESIRED;

      END;  (* End of GET SPECificationS *)

(*****************************************************************)

  PROCEDURE GETDATA;

      VAR
          OPT1,                           (* Menu options          *)
          OPT2:
                    CHAR;
          NAME:                           (* Field name            *)
                    STRING;
          DONE,                           (* Completion indicators *)
          FINISHED:
                    BOOLEAN;

(*****************************************************************)
(*                      Internal Procedures                      *)
(*****************************************************************)

      PROCEDURE PRINTHEADING;

          BEGIN
              FINISHED:=FALSE;
              WRITELN(CHR(12),' ':30,CHR(15),' DATA ENTRY ',
                      CHR(14));
              NUMREC:=NUMREC+1;

              IF (NUMREC=MAXREC) THEN     (* Data file full        *)
                  BEGIN
                      GOTOXY(0,20);
                      WRITELN(CHR(7),CHR(15),' WARNING: This is ',
                              'the last data entry you can make' ',
                               CHR(14));
                      END;

                  GOTOXY(6,3);
                  WRITELN(CHR(15),'RECORD # ',NUMREC,CHR(14));
                  GOTOXY(0,5);
                  WRITELN(' FIELD','NAME':14,'MAX WIDTH':11,'VALUE':7);
                  WRITELN(' -----','----':14,'---------':11,'-----':7);
                  NAME:=SPECS1[1];
                  IF (LENGTH(NAME)>15) THEN
                      NAME:=COPY(NAME,1,15);
                  WRITE('1':4,NAME:16,SPECS2[1]:7);
                  GOTOXY(55,7);
                  WRITE(CHR(15),'<= 9999 to stop',CHR(14));
              END;

(*************************************************************)

          PROCEDURE PICKOPTION;

              BEGIN
                  ERASE(20,3);
                  GOTOXY(0,20);
                  WRITELN('Select desired option:');
                  WRITELN('        1 - Change a value');
                  WRITELN('        2 - Enter next record');
                  GETOPTION(OPT1);
                  WHILE (OPT1<>'1') AND (OPT1<>'2') DO
                      GETOPTION(OPT1);
                  ERASE(20,3);

                  IF (OPT1='1') THEN
                      CHANGEVALUE
                  ELSE
                      FINISHED:=TRUE;
              END;

(*************************************************************)
(*                   Main body of GETDATA                    *)
(*************************************************************)

          BEGIN
          (*$I-*)
              NUMREC:=0;
              DONE:=FALSE;

              REPEAT
                  PRINTHEADING;

                  ROW:=7;
                  GOTOXY(33,ROW);
                  RESET(INPUT);
                  READ(VALUE);

                  WHILE (IORESULT=14) DO
                      HANDLEINVALID;

                  IF (VALUE=9999.0) THEN
                      BEGIN              (* Completion indicator    *)
```

```
                        DONE:=TRUE;
                        NUMREC:=NUMREC-1;
                        SPECS2[-1]:=NUMREC;
                    END;

                IF NOT(DONE) THEN           (* Record Entry                *)
                    BEGIN
                        GOTOXY(55,9);
                        WRITE(CHR(29));

                        DATA[NUMREC,1]:=VALUE;

                        FOR FIELD:=2 TO WIDTH DO
                            ENTERVALUE;
                    END;   (* End of Record Entry *)

                WHILE (NOT(FINISHED)) AND (NOT(DONE)) DO
                    PICKOPTION;

            UNTIL (DONE) OR (NUMREC=MAXREC);         (* End of REPEAT *)
        (*$I+*)                              (* Turn on I/O self check *)
        END;   (* End of GET DATA *)

(******************************************************************)
(*                      Main body of GATHERDATA                   *)
(******************************************************************)

    BEGIN
        WRITELN(CHR(12),' ':30,CHR(15),' DATA ENTRY ',CHR(14));
        GOTOXY(0,5);
        WRITELN(CHR(15),' WARNING:',CHR(14),'  Once this section ',
                'is started, data that has not been saved');
        WRITELN('          to disk via the SAVEFILE procedure ',
                'may be contaminated.');
        GOTOXY(0,10);
        WRITELN('Select desired option:');
        WRITELN('        1 - Start data entry');
        WRITELN('        2 - Exit this procedure');
        GETOPTION(OPT);
        WHILE (OPT<>'1') AND (OPT<>'2') DO
            GETOPTION(OPT);

        IF (OPT='1') THEN                             (* Accept data     *)
            BEGIN
                GETWIDTH;
                GE[illegible]PECS;
                [illegible]TA;
            END[illegible]
    END;   (* E[illegible]ER DATA *)

(************** [illegible] ****************************************)
```

```
(*$S+*)

UNIT MU_B; INTRINSIC CODE 12;

INTERFACE
    USES TRANSCEND, MAIN_UNIT;

    PROCEDURE COMPUTE(VAR DATA:RAWDATA;VAR SPECS1:HEADER1;
                      VAR SPECS2:HEADER2;INDEX:INTEGER);

    PROCEDURE RECODE(VAR DATA:RAWDATA;VAR SPECS1:HEADER1;
                     VAR SPECS2:HEADER2;INDEX:INTEGER);

IMPLEMENTATION

(*****************************************************************)
(*                     Main body of UNIT MU_B                    *)
(*****************************************************************)

PROCEDURE COMPUTE;

(*****************************************************************)
(*                                                               *)
(*         This procedure fills the indexed field in the data    *)
(*              array with a computation based on one or two     *)
(*              fields / user input constants and one operand.   *)
(*                                                               *)
(*****************************************************************)

    VAR
        I,                               (*  Iteration counter      *)
        NUMREC,                          (*  Number of records      *)
        VAR1,                            (*  1st or only variable   *)
        VAR2,                            (*  2nd variable, if req.  *)
        WIDTH:                           (*  Number of fields       *)
                 INTEGER;
        NUM1,                            (*  1st or only constant   *)
        NUM2,                            (*  2nd constant, if req.  *)
        VALUE:                           (*  Computed value         *)
                 REAL;
        OPT,                             (*  Menu options           *)
        OPT1,
        OPT2,
        OPERAND:                         (*  Selected operation     *)
                 CHAR;

(*****************************************************************)
(*                      Internal Procedures                      *)
(*****************************************************************)

    PROCEDURE PICKOPTION;

        BEGIN
            GOTOXY(0,5);
            WRITELN('This routine works by performing a computation ',
                    'based on one or two fields');
            WRITELN('and/or user input constants and one operand ',
                    '(+,-,*,/,etc.).  Any undefined');
            WRITELN('results will be stored as 99.9999.  See user''s ',
                    'guide for information on running');
            WRITELN('this procedure more than once for 2 or more ',
                    'operations.');
            GOTOXY(0,10);
            WRITELN('Select desired option:');
            WRITELN('        1 - Proceed with COMPUTE');
            WRITELN('        2 - Exit COMPUTE FIELD');
            GETOPTION(OPT);
            WHILE (OPT<>'1') AND (OPT<>'2') DO
                GETOPTION(OPT);
        END;  (* End of SELECT OPTION *)

(*******************************************************************)

    PROCEDURE USEFIELD(OPT:CHAR;VAR INDEX:INTEGER);

        BEGIN
        (*$I-*)
            IF (OPT='1') THEN
                WRITE('Enter index of 1st variable:')
            ELSE
                WRITE('Enter index of 2nd variable:');
            WRITELN(' (1 - ',WIDTH,')',CHR(13));
            RESET(INPUT);
            READ(INDEX);

            WHILE (IORESULT=14) OR (INDEX<1) OR (INDEX>WIDTH) DO
                BEGIN
                    GOTOXY(1,20);
                    WRITELN(CHR(15),'WARNING:',CHR(14),
                            '  Must be an integer between ',
                            '1 and ',WIDTH,CHR(13));
                    WRITE(' ':11,'Press any key to try again   ');
                    GOTOXY(0,20);
                    GETOPTION(OPT2);
                    ERASE(20,3);
                    ERASE(12,4);
                    GOTOXY(0,12);
                    RESET(INPUT);
                    READ(INDEX);
                END;  (* End of Bad Index *)
        (*$I+*)
        END;  (* End of USE FIELD *)

(*******************************************************************)

    PROCEDURE USENUMCONST(VAR NUM:REAL);

        BEGIN
        (*$I-*)
            WRITELN('Enter a number:',CHR(13));
            RESET(INPUT);
            READ(NUM);

            WHILE (IORESULT=14) DO
                BEGIN
                    WRITE(CHR(26));
                    GOTOXY(1,20);
                    WRITELN(CHR(15),'WARNING:',CHR(14),
                            '  Must be a number.',CHR(13));
                    WRITE(' ':11,'Press any key to try again');
                    GOTOXY(0,20);
                    GETOPTION(OPT2);
                    ERASE(20,3);
                    GOTOXY(0,12);
                    RESET(INPUT);
                    READ(NUM);
                 END;          (* End of Bad Number *)
        (*$I+*)
        END;        (* End of USE NUMber or CONSTant *)

(*****************************************************************)

    PROCEDURE SELECTOPERAND;

        BEGIN
            ERASE(5,8);
            GOTOXY(0,5);
            WRITELN(CHR(15),' SELECT DESIRED OPERAND:',CHR(14));
            WRITELN;
            WRITELN('These require a second variable:');
            WRITELN('       A - Addition          (+) ');
            WRITELN('       B - Subtraction       (-) ');
            WRITELN('       C - Multiplication    (*) ');
            WRITELN('       D - Division          (/) ');
            WRITELN;
            WRITELN('These operate on the first variable:');
            WRITELN('       E - Square            (SQR) ');
            WRITELN('       F - Square Root       (SQRT) ');
            WRITELN('       G - Natural Log       (LN) ');
            WRITELN('       H - Log Base 10       (LOG) ');
            WRITELN('       I - Exponential       (EXP) ');
            WRITELN('       J - Absolute Value    (ABS) ');
            WRITELN('       K - Truncate          (TRUNC) ');
            WRITELN('       L - Round             (ROUND) ');

            GETOPTION(OPERAND);

            WHILE (OPERAND<'A') OR (OPERAND>'L') DO
               IF (OPERAND<'a') OR (OPERAND>'l') THEN
                   BEGIN
                       GOTOXY(0,5);
                       WRITELN(CHR(15),' BAD DESIGNATOR.  ',
                               'TRY AGAIN. ',CHR(14));
                       GETOPTION(OPERAND);
                   END
               ELSE                            (* Convert to capitals *)
                   OPERAND:=CHR(ORD(OPERAND)-32);

            ERASE(5,18);
            GOTOXY(0,5);
        END;

(*****************************************************************)

    PROCEDURE TWOVAR;

        BEGIN
            IF (VAR1>0) THEN
                WRITE(SPECS1[VAR1],'   ')
            ELSE
```

```
                WRITE(NUM1:4:4,'    ');

            CASE (OPERAND) OF
                'A':  WRITE('+    ');
                'B':  WRITE('-    ');
                'C':  WRITE('*    ');
                'D':  WRITE('/    ');
            END;         (* End of CASE *)

            IF (VAR2>0) THEN
                WRITELN(SPECS1[VAR2])
            ELSE
                WRITELN(NUM2:4:4);
        END;  (* End of TWO VARiable compute *)

(***************************************************************)

    PROCEDURE ONEVAR;

        BEGIN
            CASE (OPERAND) OF
                'E':  WRITE('SQR (');
                'F':  WRITE('SQRT (');
                'G':  WRITE('LN (');
                'H':  WRITE('LOG (');
                'I':  WRITE('EXP (');
                'J':  WRITE('ABS (');
                'K':  WRITE('TRUNC (');
                'L':  WRITE('ROUND (');
            END;       (* End of CASE *)

            IF (VAR1>0) THEN
                WRITELN(SPECS1[VAR1],')')
            ELSE
                WRITELN(NUM1:4:4,' )');
        END;  (*  End of ONE VARiable compute *)

(***************************************************************)

    PROCEDURE DOCOMPUTE;

        BEGIN
        (*$R TRANSCEND *)                          (* Retain UNIT in memory *)

            GOTOXY(0,20);
            WRITE('Computing. . .    ');
            FOR I:=1 TO NUMREC DO
                BEGIN
                    IF (VAR1>0) THEN
                        NUM1:=DATA[I,VAR1];
                    IF (VAR2>0) THEN
                        NUM2:=DATA[I,VAR2];

                    CASE (OPERAND) OF
                        'A': VALUE:=NUM1+NUM2;
                        'B': VALUE:=NUM1-NUM2;
                        'C': VALUE:=NUM1*NUM2;
                        'D': IF (NUM2=0.0) THEN
                                 VALUE:=99.9999
                             ELSE
```

```
                                    VALUE:=NUM1/NUM2;
                               'E': VALUE:=SQR(NUM1);
                               'F': IF (NUM1<0.0) THEN
                                       VALUE:=99.9999
                                    ELSE
                                       VALUE:=SQRT(NUM1);
                               'G': IF (NUM1<=0.0) THEN
                                       VALUE:=99.9999
                                    ELSE
                                       VALUE:=LN(NUM1);
                               'H': IF (NUM1<=0.0) THEN
                                       VALUE:=99.9999
                                    ELSE
                                       VALUE:=LOG(NUM1);
                               'I': VALUE:=EXP(NUM1);
                               'J': VALUE:=ABS(NUM1);
                               'K': VALUE:=TRUNC(NUM1);
                               'L': VALUE:=ROUND(NUM1);
                            END;     (*  End of CASE *)

                            DATA[I,INDEX]:=VALUE;
                        END;
                END;        (* End of DO the COMPUTE *)

(*****************************************************************)
(*                        Main body of COMPUTE                   *)
(*****************************************************************)

    BEGIN
        NUMREC:=SPECS2[-1];                (* Initialize parameters *)
        WIDTH:=SPECS2[0];
        VAR1:=0;
        VAR2:=0;

        WRITELN(CHR(12),' ':25,CHR(15),' COMPUTE FIELD ROUTINE ',
                CHR(14));

        PICKOPTION;

        IF (OPT='1') THEN                  (* Proceed with Compute  *)
            BEGIN
                ERASE(5,8);
                GOTOXY(0,5);
                WRITELN('Select desired option:');
                WRITELN('       1 - Identify field of 1st variable');
                WRITELN('       2 - Enter a number (constant)');
                GETOPTION(OPT1);
                WHILE (OPT1<>'1') AND (OPT1<>'2') DO
                    GETOPTION(OPT1);

                GOTOXY(0,10);                (* Get first variable  *)
                IF (OPT1='1') THEN
                    USEFIELD(OPT1,VAR1)
                ELSE
                    USENUMCONST(NUM1);

                SELECTOPERAND;

                IF (OPERAND<'E') THEN        (* Get second variable *)
                    BEGIN
                    WRITELN('Select desired option:');
                    WRITELN('        1 - Identify field of ',
                            '2nd variable');
                    WRITELN('        2 - Enter a number');
                    GETOPTION(OPT2);
                    WHILE (OPT2<>'1') AND (OPT2<>'2') DO
                        GETOPTION(OPT2);

                    GOTOXY(0,10);
                    IF (OPT2='1') THEN
                        USEFIELD('2',VAR2)
                    ELSE
                        USENUMCONST(NUM2);

                END;  (* End of Get Second Variable *)

            ERASE(5,8);
            GOTOXY(0,5);
            WRITELN(CHR(15),' COMPUTATION SELECTED:',CHR(14));
            GOTOXY(25,10);

            IF (OPERAND<'E') THEN        (* Display computation *)
                TWOVAR
            ELSE
                ONEVAR;

            GOTOXY(0,18);
            WRITELN('Select desired option:');
            WRITELN('        1 - Proceed with COMPUTE');
            WRITELN('        2 - Skip this COMPUTE');
            GETOPTION(OPT);
            WHILE (OPT<>'1') AND (OPT<>'2') DO
                GETOPTION(OPT);

            ERASE(5,16);
            IF (OPT='1') THEN
                DOCOMPUTE;

        END;  (* End of Proceed with Compute *)

    END;  (* End of COMPUTE *)

(**********************************************************************)

(*$I PSPP:RECODE *)                     (* Include procedure in UNIT *)

(**********************************************************************)
(*                    Initialization part of UNIT                     *)
(**********************************************************************)

END.
```

```
PROCEDURE RECODE;

(***************************************************************)
(*                                                             *)
(*       This procedure fills the indexed field in the data    *)
(*            array with user input constants based on         *)
(*            partitions within that or a different field.     *)
(*                                                             *)
(***************************************************************)

    TYPE
        BUFFER=ARRAY[1..MAXREC] OF REAL;

    VAR
        FIELD,                          (* Field recoding is based on *)
        I,                              (* Iteration counter          *)
        NUMREC,                         (* Number of records in file  *)
        WIDTH:                          (* Number of fields in file   *)
                  INTEGER;
        BOTTOM,                         (* Bottom edge of partition   *)
        TOP,                            (* Top edge of partition      *)
        VALUE:                          (* Recoded value in partition *)
                  REAL;
        NEWFIELD:                       (* Temporary recoded field    *)
                  BUFFER;
        OPT0,                           (* Menu options               *)
        OPT1,
        OPT2,
        OPT3,
        EXTREME:                        (* End point indicator        *)
                  CHAR;
        DONE,                           (* Completion indicator       *)
        HIGHEST,                        (* High end point used        *)
        LOWEST:                         (* Low end point used         *)
                  BOOLEAN;

(***************************************************************)
(*                    Internal Procedures                      *)
(***************************************************************)

    PROCEDURE DISPLAYINSTRUCTIONS;

        BEGIN
            WRITELN(CHR(12),' ':25,CHR(15),' RECODE FIELD ROUTINE ',
                    CHR(14));
            GOTOXY(0,5);
            WRITELN('This routine works by partitioning the data ',
                    'of a specified field based');
            WRITELN('on range(s) between two endpoints.  You ',
                    'have the option of entering');
            WRITELN('numeric endpoints or using the values ',
                    'LOWEST and HIGHEST.  Those points');
            WRITELN('indicate the two extremes of the data ',
                    'field.',CHR(13));
            WRITELN('NOTE:  Once started, you can not leave ',
                    'this routine without using LOWEST');
            WRITELN('and HIGHEST at least once.  See user''s ',
                    'guide for further information.');
            GOTOXY(0,20);
            WRITELN('Select desired option:');
            WRITELN('        1 - Proceed with RECODE');
            WRITELN('        2 - Exit RECODE FIELD');
            GETOPTION(OPTO);
            WHILE (OPTO<>'1') AND (OPTO<>'2') DO
                GETOPTION(OPTO);
            ERASE(5,7);
            ERASE(20,3);
        END;  (* End of DISPLAY INSTRUCTIONS *)

(******************************************************************)

    PROCEDURE GETFIELD;

        BEGIN
        (*$I-*)
            GOTOXY(0,5);
            WRITELN('Enter field to use in recoding: (1 - ',
                     WIDTH,')',CHR(13));
            RESET(INPUT);
            READ(FIELD);

            WHILE (IORESULT=14) OR (FIELD<1) OR (FIELD>WIDTH) DO
                BEGIN
                    GOTOXY(1,20);
                    WRITELN(CHR(15),'WARNING:',CHR(14),'  Bad ',
                           'index.  Enter an integer between ',
                           '1 and ',WIDTH,CHR(13));
                    WRITELN(' ':11,'Press any key to continue');
                    GOTOXY(0,20);
                    GETOPTION(OPT2);
                    ERASE(7,16);
                    GOTOXY(0,7);
                    RESET(INPUT);
                    READ(FIELD);
                END;
        (*$I+*)
        END;  (* End of GET FIELD *)

(******************************************************************)

    PROCEDURE GETOPT1;

        BEGIN
            ERASE(5,3);
            GOTOXY(0,5);
            WRITELN('Select desired option:');
            WRITELN('        1 - Enter a partition');
            WRITELN('        2 - Exit RECODE FIELD');
            GETOPTION(OPT1);
            WHILE (OPT1<>'1') AND (OPT1<>'2') DO
                GETOPTION(OPT1);
            EXTREME:=' ';
        END;

(******************************************************************)

    PROCEDURE GETBOTTOMOPTION;

        BEGIN
```

```
          ERASE(5,3);
          GOTOXY(0,5);
          WRITELN('Set partition bottom edge using:');
          WRITELN('     1 - Numeric endpoint');
          WRITELN('     2 - LOWEST value');
          GETOPTION(OPT2);
          WHILE (OPT2<>'1') AND (OPT2<>'2') DO
              GETOPTION(OPT2);
      END;

(*******************************************************************)

  PROCEDURE GETTOPOPTION;

      BEGIN
          ERASE(5,3);
          GOTOXY(0,5);
          WRITELN('Set partition top edge using:');
          WRITELN('     1 - Numeric endpoint');
          WRITELN('     2 - HIGHEST value');
          GETOPTION(OPT2);
          WHILE (OPT2<>'1') AND (OPT2<>'2') DO
              GETOPTION(OPT2);
      END;

(*******************************************************************)

  PROCEDURE NUMERICBOTTOM;

      BEGIN
      (*$I-*)
          GOTOXY(0,10);
          WRITELN('Enter lower endpoint:',CHR(13));
          RESET(INPUT);
          READ(BOTTOM);

          WHILE (IORESULT=14) DO
              BEGIN
                  GOTOXY(1,20);
                  WRITELN(CHR(15),'WARNING:',CHR(14),'  Must ',
                          'be a number.',CHR(13));
                  WRITELN(' ':11,'Press any key to try again.');
                  GOTOXY(0,20);
                  GETOPTION(OPT3);
                  ERASE(12,11);
                  GOTOXY(0,12);
                  RESET(INPUT);
                  READ(BOTTOM);
              END;  (* End of Bad Bottom *)

          ERASE(10,3);
      (*$I+*)
      END;  (* End of NUMERIC BOTTOM *)

(*******************************************************************)

  PROCEDURE NUMERICTOP;

      BEGIN
      (*$I-*)
```

```
            GOTOXY(0,10);
            WRITELN('Enter upper endpoint:',CHR(13));
            RESET(INPUT);
            READ(TOP);

            WHILE (IORESULT=14) OR (TOP<BOTTOM) DO
                BEGIN
                    GOTOXY(1,20);
                    WRITELN(CHR(15),'WARNING:',CHR(14),'  ',
                        'Must be a number greater than ',
                         BOTTOM:6:5,CHR(13));
                    WRITELN(' ':11,'Press any key to try again.');
                    GOTOXY(0,20);
                    GETOPTION(OPT3);
                    ERASE(12,11);
                    GOTOXY(0,12);
                    RESET(INPUT);
                    READ(TOP);
                END;  (* End of Bad Top *)

            ERASE(10,3);
        (*$I+*)
        END;  (* End of NUMERIC TOP *)

(*********************************************************************)

    PROCEDURE GETRECODER;

        BEGIN
        (*$I-*)
            WRITELN('Enter value to recode partition with:');
            ERASE(6,2);
            RESET(INPUT);
            READ(VALUE);

            WHILE (IORESULT=14) DO
                BEGIN
                    WRITE(CHR(26));
                    GOTOXY(1,20);
                    WRITELN(CHR(15),'WARNING:',CHR(14),'Must be ',
                            'a number.',CHR(13));
                    WRITELN(' ':11,'Press any key to try again.');
                    GOTOXY(0,20);
                    GETOPTION(OPT3);
                    ERASE(20,3);
                    GOTOXY(0,8);
                    RESET(INPUT);
                    READ(VALUE);
                END;  (* End of Bad Value *)
        (*$I+*)
        END;  (* End of GET RECODER *)
(*********************************************************************)

    PROCEDURE VIEWRECODE;

        BEGIN
            ERASE(5,4);
            WRITELN('Partition is:',CHR(13));
            WRITE('                Recode ');
            IF (EXTREME='L') OR (EXTREME='B') THEN
            WRITE('LOWEST')
        ELSE
            WRITE(BOTTOM:6:4);
        WRITE(' to ');
        IF (EXTREME='H') OR (EXTREME='B') THEN
            WRITE('HIGHEST')
        ELSE
            WRITE(TOP:6:4);
        WRITELN(' with ',VALUE:6:4,CHR(13),CHR(13));

        WRITELN('Select desired option:');
        WRITELN('        1 - Proceed with RECODE');
        WRITELN('        2 - Skip this RECODE');
        GETOPTION(OPT3);
        WHILE (OPT3<>'1') AND (OPT3<>'2') DO
            GETOPTION(OPT3);
        ERASE(14,3);
    END;  (* End of VIEW RECODE *)

(*********************************************************************)

PROCEDURE DORECODE;

    BEGIN
        GOTOXY(0,20);
        WRITE('Recoding. . .    ');

        CASE (EXTREME) OF
         ' ': FOR I:=1 TO NUMREC DO
                  IF (DATA[I,FIELD]>BOTTOM) AND
                     (DATA[I,FIELD]<=TOP) THEN
                         NEWFIELD[I]:=VALUE;
         'L': FOR I:=1 TO NUMREC DO
                  IF (DATA[I,FIELD]<=TOP) THEN
                         NEWFIELD[I]:=VALUE;
         'H': FOR I:=1 TO NUMREC DO
                  IF (DATA[I,FIELD]>BOTTOM) THEN
                         NEWFIELD[I]:=VALUE;
         'B': FOR I:=1 TO NUMREC DO
                         NEWFIELD[I]:=VALUE;
        END;  (* End of CASE *)

        ERASE(20,1);
    END;  (* End of DO the RECODE *)

(*********************************************************************)

PROCEDURE MAKESAVEFINAL;

    BEGIN
        ERASE(5,3);
        GOTOXY(0,5);
        WRITELN('Select desired option:');
        WRITELN('        1 - Save the RECODE');
        WRITELN('        2 - Exit without saving');
        GETOPTION(OPT2);
        WHILE (OPT2<>'1') AND (OPT2<>'2') DO
            GETOPTION(OPT2);

        IF (OPT2='1') THEN
```

```
          BEGIN
              GOTOXY(0,15);
              WRITELN('Saving. . .');
              FOR I:=1 TO NUMREC DO
                  DATA[I,INDEX]:=NEWFIELD[I];
          END;
      END;  (* End of MAKE SAVE FINAL *)

(*****************************************************************)
(*                       Main body of RECODE                      *)
(*****************************************************************)

  BEGIN
      NUMREC:=SPECS2[-1];            (* Initialize parameters     *)
      WIDTH:=SPECS2[0];
      DONE:=FALSE;
      HIGHEST:=FALSE;
      LOWEST:=FALSE;

      DISPLAYINSTRUCTIONS;

      IF (OPT0='1') THEN             (* Get field to partition on  *)
          GETFIELD
      ELSE
          DONE:=TRUE;

      WHILE NOT(DONE) DO                    (* Do a Recode          *)
          BEGIN
              GETOPT1;

              IF (OPT1='1') THEN            (* Get partition range *)
                  BEGIN
                      GETBOTTOMOPTION;

                      IF (OPT2='1') THEN       (* Numeric bottom *)
                          NUMERICBOTTOM
                      ELSE
                          BEGIN                (* LOWEST bottom  *)
                              EXTREME:='L';
                              BOTTOM:=-MAXINT;
                              LOWEST:=TRUE;
                          END;

                      GETTOPOPTION;

                      IF (OPT2='1') THEN       (* Numeric top    *)
                          NUMERICTOP
                      ELSE
                          BEGIN                (* HIGHEST top    *)
                              IF (EXTREME='L') THEN
                                  EXTREME:='B'
                              ELSE
                                  EXTREME:='H';
                              TOP:=MAXINT;
                              HIGHEST:=TRUE;
                          END;

                      GOTOXY(0,5);             (* Recode value   *)
                      GETRECODER;
```

```
                    GOTOXY(0,5);                  (* Display recode *)
                    VIEWRECODE;

                    IF (OPT3='1') THEN
                        DORECODE;

                    ERASE(5,8);
                END
            ELSE
                IF (HIGHEST) AND (LOWEST) THEN
                    DONE:=TRUE
                ELSE
                    BEGIN
                        GOTOXY(1,20);
                        WRITELN(CHR(15),'WARNING:',CHR(14),
                               '  Must reference both HIGHEST ',
                               'and LOWEST once each.',CHR(13));
                        WRITELN(' ':11,'Press any key to ',
                                'continue');
                        GOTOXY(0,20);
                        GETOPTION(OPT2);
                        ERASE(20,3);
                    END;
        END;  (* End of Do a Recode *)

    IF (OPTO<>'2') THEN
        MAKESAVEFINAL;

END;  (* End of RECODE *)

(*****************************************************************)
```

```
(*$S+*)

UNIT MU_C; INTRINSIC CODE 13;

INTERFACE
    USES MAIN_UNIT;

    PROCEDURE HANDLEINVALID(VAR VALUE:REAL;ROW:INTEGER);

    PROCEDURE ENTERVALUE(VAR DATA:RAWDATA;VAR SPECS1:HEADER1;
                         VAR SPECS2:HEADER2;INDEX,FIELD:INTEGER);

    PROCEDURE CHANGEVALUE(VAR DATA:RAWDATA;INDEX,WIDTH:INTEGER);

    PROCEDURE SUBARECORD(VAR DATA:RAWDATA;VAR SPECS1:HEADER1;
                         VAR SPECS2:HEADER2);

    PROCEDURE ADDARECORD(VAR DATA:RAWDATA;VAR SPECS1:HEADER1;
                         VAR SPECS2:HEADER2);

    PROCEDURE CHGARECORD(VAR DATA:RAWDATA;VAR SPECS1:HEADER1;
                         VAR SPECS2:HEADER2);

    PROCEDURE SUBAFIELD(VAR DATA:RAWDATA:VAR SPECS1:HEADER1;
                        VAR SPECS2:HEADER2);

IMPLEMENTATION

(*********************************************************************)
(*                       Main body of MU_C                          *)
(*********************************************************************)

PROCEDURE HANDLEINVALID;

    VAR OPT:  CHAR;                                  (* Menu option  *)

    BEGIN
    (*$I-*)
        GOTOXY(1,20);
        WRITELN(CHR(15),'WARNING:',CHR(14),'  Value must ',
                'be a number. Press any key to continue');
        GOTOXY(0,20);
        GETOPTION(OPT);
        ERASE(20,1);
        GOTOXY(33,ROW);
        WRITELN(CHR(29));
        GOTOXY(33,ROW);
        RESET(INPUT);
        READ(VALUE);
    (*$I+*)
    END;  (* End of HANDLEINVALID *)

(*********************************************************************)

PROCEDURE ENTERVALUE;

    VAR
        ROW:                              (* Row on screen          *)
                INTEGER;
        VALUE:                            (* Data value as entered *)
```

```
                  REAL;
          NAME:                                    (* Field name             *)
                  STRING;

      BEGIN
      (*$I-*)
          ROW:=FIELD+8;
          GOTOXY(0,ROW);
          NAME:=SPECS1[FIELD];
          IF (LENGTH(NAME)>15) THEN                (* Truncate names to fit *)
              NAME:=COPY(NAME,1,15);
          WRITELN(FIELD:4,NAME:16,SPECS2[FIELD]:7);

          GOTOXY(33,ROW);
          RESET(INPUT);
          READ(VALUE);

          WHILE (IORESULT=14) DO
              HANDLEINVALID(VALUE,ROW);

          DATA[INDEX,FIELD]:=VALUE;
      (*$I+*)
      END;  (* End of ENTERVALUE *)

(********************************************************************)

PROCEDURE CHANGEVALUE;

      VAR
          FIELD,                               (* Field to change           *)
          ROW:                                 (* Row on screen             *)
                  INTEGER;
          VALUE:                               (* Data value to be stored   *)
                  REAL;
          OPT:                                 (* Menu option               *)
                  CHAR;

      BEGIN
      (*$I-*)
          ERASE(19,4);
          GOTOXY(0,19);
          WRITELN('Enter field to change: (1 - ',WIDTH,')');
          WRITELN('       (0 = Skip change)');
          RESET(INPUT);
          READ(FIELD);

          WHILE (IORESULT=14) OR (FIELD<0) OR (FIELD>WIDTH) DO
              BEGIN
                  GOTOXY(0,22);
                  WRITE(CHR(7),'Bad field number.  Press any ',
                          'key to try again   ');
                  GETOPTION(OPT);
                  ERASE(21,2);
                  GOTOXY(0,21);
                  RESET(INPUT);
                  READ(FIELD);
              END; (* End of Invalid Index *)
          ERASE(19,4);

          IF (FIELD<>0) THEN                              (* Make change *)
          BEGIN
               ROW:=FIELD+8;
               GOTOXY(33,ROW);
               WRITE(' ':23,CHR(15),'<= Enter new value',CHR(14));

               GOTOXY(33,ROW);
               RESET(INPUT);
               READ(VALUE);

               WHILE (IORESULT=14) DO
                    HANDLEINVALID(VALUE,ROW);;

               DATA[INDEX,FIELD]:=VALUE;
               GOTOXY(56,ROW);
               WRITELN(CHR(29));
          END;
     (*$I+*)
     END;  (* End of CHANGEVALUE *)

(*********************************************************************)

PROCEDURE SUBARECORD;

(*********************************************************************)
(*                                                                   *)
(*        This procedure removes one record from a file by          *)
(*              overwriting it with the last record in the          *)
(*              file and decrementing NUMREC; the Number of         *)
(*              Records counter stored in SPECS2[-1].               *)
(*                                                                   *)
(*********************************************************************)

     VAR
          I,                                 (* Iteration counter     *)
          INDEX,                             (* Record to remove      *)
          NUMREC,                            (* Number of records     *)
          WIDTH:                             (* Number of fields      *)
               INTEGER;
          OPT1,                              (* Menu options          *)
          OPT2:
               CHAR;

(*********************************************************************)
(*                        Internal Procedures                        *)
(*********************************************************************)

     PROCEDURE BADINDEX;

          BEGIN
          (*$I-*)
               GOTOXY(1,20);
               WRITELN(CHR(15),'WARNING:',CHR(14),'  Bad index. ',
                       'Enter an integer between 1 and ',
                       NUMREC,'.',CHR(13));
               WRITELN(' ':11,'Press any key to try again.');
               GOTOXY(0,20);
               GETOPTION(OPT2);
               ERASE(8,15);
               GOTOXY(0,8);
               RESET(INPUT);
```

```
            READ(INDEX);
        (*$I+*)
        END;   (* End of Bad Index *)

(*******************************************************************)

    PROCEDURE VIEWRECORD;

        BEGIN
            GOTOXY(0,8);
            WRITE(CHR(15),' RECORD # ',INDEX,' ',CHR(14));
            GOTOXY(0,10);
            FOR I:=1 TO WIDTH DO              (* Display field names *)
                WRITE(SPECS1[I]:SPECS2[I],' ');
            WRITELN;
            FOR I:=1 TO WIDTH DO              (* Display the record   *)
                WRITE(DATA[INDEX,I]:SPECS2[I]:4,' ');
            WRITELN;
        END;

(*******************************************************************)

    PROCEDURE DOREMOVE;

        BEGIN
            FOR I:=1 TO WIDTH DO
                DATA[INDEX,I]:=DATA[NUMREC,I];
            SPECS2[-1]:=NUMREC-1;
        END;   (* End of Do Remove *)

(*******************************************************************)
(*                    Main body of SUBARECORD                      *)
(*******************************************************************)

    BEGIN
    (*$I-*)
        NUMREC:=SPECS2[-1];                 (* Initialize parameters *)
        WIDTH:=SPECS2[0];

        WRITELN(CHR(12),' ':24,CHR(15),' REMOVE RECORD ROUTINE ',
                CHR(14));
        GOTOXY(0,5);
        WRITELN('Select desired option:');
        WRITELN('        1 - Proceed with REMOVE');
        WRITELN('        2 - Exit REMOVE RECORD');
        GETOPTION(OPT1);
        WHILE (OPT1<>'1') AND (OPT1<>'2') DO
            GETOPTION(OPT1);
        ERASE(5,3);

        IF (OPT1='1') AND (NUMREC>0) THEN (* Proceed to Remove     *)
            BEGIN
                GOTOXY(0,5);
                WRITELN('Enter index of record to be removed: ',
                        '(1 - ',NUMREC,')');
                WRITELN(CHR(7),CHR(13));
                RESET(INPUT);
                READ(INDEX);

                WHILE (IORESULT=14) OR (INDEX<1) OR
```

```
                                    (INDEX>NUMREC) DO
                BADINDEX;

            ERASE(5,4);
            VIEWRECORD;

            GOTOXY(0,14);
            WRITELN('Select desired option:');
            WRITELN('        1 - Proceed with REMOVE');
            WRITELN('        2 - Cancel the REMOVE');
            GETOPTION(OPT2);
            WHILE (OPT2<>'1') AND (OPT2<>'2') DO
                GETOPTION(OPT2);

            IF (OPT2='1') THEN
                DOREMOVE;

        END;  (* End of Proceed to Remove *)
    (*$I+*)
    END;  (*  End of SUB A RECORD *)

(**********************************************************************)

PROCEDURE ADDARECORD;

(**********************************************************************)
(*                                                                    *)
(*          This procedure adds one record to a file at the          *)
(*               end, if there is room, and updates NUMREC;           *)
(*               the Number of Records counter stored in              *)
(*               SPECS2[-1].                                          *)
(*                                                                    *)
(**********************************************************************)

    VAR
        INDEX,                                (* Iteration counter      *)
        NUMREC,                               (* Number of records      *)
        ROOM,                                 (* Available room         *)
        ROW,                                  (* Row on screen          *)
        WIDTH:                                (* Number of fields       *)
               INTEGER;
        VALUE:                                (* Value input by user    *)
               REAL;
        OPT1,                                 (* Menu options           *)
        OPT2:
               CHAR;
        DONE:                                 (* Completion indicator   *)
               BOOLEAN;

(**********************************************************************)
(*                         Internal Procedures                        *)
(**********************************************************************)

    PROCEDURE INPUTRECORD;

        BEGIN
            NUMREC:=NUMREC+1;
            GOTOXY(5,5);
            WRITELN(CHR(15),' RECORD # ',NUMREC,' ',CHR(14));
            WRITELN;
```

```
          WRITELN(' FIELD','NAME':14,'MAX WIDTH':11,'VALUE':7);
          WRITELN(' -----','----':14,'---------':11,'-----':7);

          FOR INDEX:=1 TO WIDTH DO
              ENTERVALUE(DATA,SPECS1,SPECS2,NUMREC,INDEX);

          SPECS2[-1]:=NUMREC;
      END;  (* End of INPUT the RECORD *)

(*****************************************************************)

    PROCEDURE MAKECORRECT;

      BEGIN
          GOTOXY(0,20);
          WRITELN('Select desired option:');
          WRITELN('        1 - Change a value');
          WRITELN('        2 - Exit ADD RECORD');
          GETOPTION(OPT1);
          WHILE (OPT1<>'1') AND (OPT1<>'2') DO
              GETOPTION(OPT1);
          ERASE(20,3);

          IF (OPT1='1') THEN
              CHANGEVALUE(DATA,NUMREC,WIDTH)
          ELSE
              DONE:=TRUE;
      END;  (* End of MAKE CORRECTions *)

(*****************************************************************)
(*                      Main body of ADDARECORD                  *)
(*****************************************************************)

    BEGIN
      NUMREC:=SPECS2[-1];               (* Initialize parameters *)
      WIDTH:=SPECS2[0];
      ROOM:=MAXREC-NUMREC;
      DONE:=FALSE;

      WRITELN(CHR(12),' ':26,CHR(15),' ADD RECORD ROUTINE ',
              CHR(14));
      GOTOXY(0,5);

      IF (ROOM=0) THEN                  (* No room to add record *)
          BEGIN
              GOTOXY(1,21);
              WRITELN(CHR(15),'WARNING:',CHR(14),'  The file ',
                     'is full; no more records can be added.');
              WRITELN(' ':11,'Press any key to continue');
              GOTOXY(0,21);
              GETOPTION(OPT1);
          END
      ELSE
          BEGIN                         (* Room available to add *)
              WRITELN('Select desired option:');
              WRITELN('        1 - Proceed with ADD');
              WRITELN('        2 - Exit ADD RECORD');
              GETOPTION(OPT1);
              WHILE (OPT1<>'1') AND (OPT1<>'2') DO
                  GETOPTION(OPT1);
```

```
                ERASE(5,3);

                IF (OPT1='1') THEN
                    INPUTRECORD
                ELSE
                    DONE:=TRUE;

                WHILE NOT(DONE) DO
                    MAKECORRECT;

            END;  (* End of Room Available *)

    END;  (* End of ADD A RECORD *)

(*********************************************************************)

PROCEDURE CHGARECORD;

(*********************************************************************)
(*                                                                   *)
(*          This procedure changes the contents of one record       *)
(*               in the data file by overwriting the old             *)
(*               contents with user inputs.                          *)
(*                                                                   *)
(*********************************************************************)

    VAR
        I,                                  (* Iteration counter      *)
        INDEX,                              (* Record being changed   *)
        NUMREC,                             (* Number of records      *)
        ROW,                                (* Row on screen          *)
        WIDTH:                              (* Number of fields       *)
                INTEGER;
        VALUE:                              (* Value as input by user *)
                REAL;
        OPT1,                               (* Menu options           *)
        OPT2:
                CHAR;
        NAME:                               (* Field name             *)
                STRING;
        DONE:                               (* Completion indicator   *)
                BOOLEAN;

(*********************************************************************)
(*                        Internal Procedures                        *)
(*********************************************************************)

     PROCEDURE BADRECORDINDEX;

         BEGIN
         (*$I-*)
             GOTOXY(1,20);
             WRITELN(CHR(15),'WARNING:',CHR(14),
                     '  Bad index.  Enter an integer ',
                     'between 1 and ',NUMREC,CHR(13));
             WRITELN(' ':11,'Press any key to try again');
             GOTOXY(0,20);
             GETOPTION(OPT2);
             ERASE(8,15);
             GOTOXY(0,8);
```

```
          RESET(INPUT);
          READ(INDEX);
      (*$I+*)
      END;   (* End of BAD RECORD INDEX *)

(*****************************************************************)

   PROCEDURE SHOWCURRENT;

      BEGIN
          ERASE(5,4);
          GOTOXY(5,5);
          WRITELN(CHR(15),' RECORD # ',INDEX,' ',CHR(14));
          GOTOXY(0,7);
          WRITELN(' FIELD','NAME':14,'MAX WIDTH':11,'VALUE':7);
          WRITELN(' -----','----':14,'---------':11,'-----':7);

          FOR I:=1 TO WIDTH DO
              BEGIN
                  NAME:=SPECS1[I];
                  IF (LENGTH(NAME)>15) THEN
                      NAME:=COPY(NAME,1,15);
                  WRITELN(I:4,NAME:16,SPECS2[I]:7,' ':7,
                          DATA[INDEX,I]:6:4);
              END;
      END;   (* End of SHOW CURRENT data *)

(*****************************************************************)

   PROCEDURE CHANGEFIELDS;

      BEGIN
          GOTOXY(0,20);
          WRITELN('Select desired option:');
          WRITELN('         1 - Change a value');
          WRITELN('         2 - Exit CHANGE RECORD');
          GETOPTION(OPT2);
          WHILE (OPT2<>'1') AND (OPT2<>'2') DO
              GETOPTION(OPT2);
          ERASE(20,3);

          IF (OPT2='1') THEN
              CHANGEVALUE(DATA,INDEX,WIDTH)
          ELSE
              DONE:=TRUE;
      END;   (* End of CHANGE the FIELDS *)

(*****************************************************************)
(*                    Main body of CHGARECORD                    *)
(*****************************************************************)

  BEGIN
  (*$I-*)
     NUMREC:=SPECS2[-1];                  (* Initialize parameters *)
     WIDTH:=SPECS2[0];
     DONE:=FALSE;

     WRITELN(CHR(12),' ':24,CHR(15),' CHANGE RECORD ROUTINE ',
             CHR(14));
     GOTOXY(0,5);
```

```
            WRITELN('Select desired option:');
            WRITELN('         1 - Proceed with CHANGE');
            WRITELN('         2 - Exit CHANGE RECORD');
            GETOPTION(OPT1);
            WHILE (OPT1<>'1') AND (OPT1<>'2') DO
                GETOPTION(OPT1);
            ERASE(5,3);

            IF (OPT1='1') AND (NUMREC>0) THEN      (* Proceed to change *)
                BEGIN
                    GOTOXY(0,5);
                    WRITELN('Enter record to change: (1 - ',NUMREC,')');
                    WRITELN('         (0 = Skip change)');
                    WRITELN;
                    RESET(INPUT);
                    READ(INDEX);

                    WHILE (IORESULT=14) OR (INDEX<0) OR (INDEX>NUMREC) DO
                        BADRECORDINDEX;

                    IF (INDEX<>0) THEN              (* Do the changes    *)
                        BEGIN
                            SHOWCURRENT;

                            WHILE NOT(DONE) DO
                                CHANGEFIELDS;
                        END;

                END;  (* End of Proceed to Change *)
        (*$I+*)
        END;  (* End of CHG A RECORD *)

(***************************************************************)

(*$I P6PP:SUBAFLD *)                 (* Include procedure in UNIT *)

(***************************************************************)
(*                   Initialization part of UNIT              *)
(***************************************************************)

END.
```

```
PROCEDURE SUBAFIELD;

(****************************************************************)
(*                                                              *)
(*       This procedure removes a field or variable from a      *)
(*            file by overwriting it with the last field in     *)
(*            the file and decrementing WIDTH; the Number       *)
(*            of Fields counter stored in SPECS2[0].            *)
(*                                                              *)
(****************************************************************)

    VAR
        I,                                 (* Iteration counter        *)
        INDEX,                             (* Field to remove          *)
        NUMREC,                            (* Number of records        *)
        WIDTH:                             (* Number of fields         *)
                INTEGER;
        OPT1,                              (* Menu options             *)
        OPT2:
                CHAR;

(****************************************************************)
(*                     Internal Procedures                      *)
(****************************************************************)

    PROCEDURE BADINDEX;

        BEGIN
        (*$I-*)
            GOTOXY(1,22);
            WRITE(CHR(15),'WARNING:',CHR(14),'  Bad field number.',
                    '  Press any key to try again.');
            GOTOXY(0,22);
            GETOPTION(OPT2);
            ERASE(20,3);
            GOTOXY(0,20);
            RESET(INPUT);
            READ(INDEX);
        (*$I+*)
        END;  (* End of BAD INDEX *)

(****************************************************************)

    PROCEDURE DOREMOVAL;

        BEGIN
            FOR I:=1 TO NUMREC DO
                DATA[I,INDEX]:=DATA[I,WIDTH];
            SPECS1[INDEX]:=SPECS1[WIDTH];
            SPECS2[INDEX]:=SPECS2[WIDTH];
            SPECS2[0]:=WIDTH-1;
        END;  (* End of DO REMOVAL *)

(****************************************************************)
(*                     Main body of SUBAFIELD                   *)
(****************************************************************)

    BEGIN
    (*$I-*)
```

```
          NUMREC:=SPECS2[-1];                   (* Initialize parameters *)
          WIDTH:=SPECS2[0];

          WRITELN(CHR(12),' ':25,CHR(15),' REMOVE FIELD ROUTINE ',
                  CHR(14));
          GOTOXY(0,5);
          WRITELN('Select desired option:');
          WRITELN('        1 - Proceed with REMOVE');
          WRITELN('        2 - Exit REMOVE FIELD');
          GETOPTION(OPT1);
          WHILE (OPT1<>'1') AND (OPT1<>'2') DO
              GETOPTION(OPT1);
          ERASE(5,3);

          IF (OPT1='1') AND (WIDTH>0) THEN        (* Proceed to Remove *)
              BEGIN
                  GOTOXY(0,5);
                  WRITELN(' #  ',CHR(15),' FIELDS IN FILE ',CHR(14));
                  WRITELN('--- ----------------');
                  FOR INDEX:=1 TO WIDTH DO
                      WRITELN(INDEX:2,'  ',SPECS1[INDEX]);

                  GOTOXY(0,18);
                  WRITELN('Enter field to be removed: (1 - ',
                           WIDTH,')');
                  WRITELN('       (0 = Skip removal)');

                  RESET(INPUT);
                  READ(INDEX);

                  WHILE (IORESULT=14) OR (INDEX<0) OR (INDEX>WIDTH) DO
                      BADINDEX;

                  ERASE(18,5);

                  IF (INDEX<>0) THEN
                      DOREMOVAL;

              END;  (* End of Proceed to Remove *)

          IF (SPECS2[0]=0) THEN
              BEGIN
                  GOTOXY(1,22);
                  WRITELN(CHR(15),'WARNING:',CHR(14),'  File is ',
                          'now empty.  Press any key to continue.');
                  GOTOXY(0,22);
                  GETOPTION(OPT2);
              END;
     (*$I+*)
     END;  (* End of SUB A FIELD *)

(*********************************************************************)
```

```
(*$S+*)

UNIT MU_D; INTRINSIC CODE 14;

INTERFACE
    USES TRANSCEND, MAIN_UNIT, MU_B, MU_C;

    PROCEDURE DISPLAYSPECS(VAR SPECS1:HEADER1;VAR SPECS2:HEADER2);

    PROCEDURE GETFLDWIDTH(VAR SPECS2:HEADER2;VAR INDEX,ROW,
                                          COLS:INTEGER);

    PROCEDURE GETFLDNAME(VAR SPECS1:HEADER1;VAR SPECS2:HEADER2;
                         INDEX,ROW:INTEGER);

    PROCEDURE ADDAFIELD(VAR DATA:RAWDATA;VAR SPECS1:HEADER1;
                        VAR SPECS2:HEADER2);

    PROCEDURE CHGAFIELD(VAR DATA:RAWDATA;VAR SPECS1:HEADER1;
                        VAR SPECS2:HEADER2);

    PROCEDURE FILLFIELD(VAR DATA:RAWDATA;VAR SPECS1:HEADER1;
                        VAR SPECS2:HEADER2;INDEX:INTEGER);

    PROCEDURE MODIFILE(VAR DATA:RAWDATA;VAR SPECS1:HEADER1;
                                        VAR SPECS2:HEADER2);

    PROCEDURE USERINPUT(VAR DATA:RAWDATA;VAR SPECS1:HEADER1;
                        VAR SPECS2:HEADER2;INDEX:INTEGER);

IMPLEMENTATION

(*******************************************************************)
(*                      Main body of MU_D                          *)
(*******************************************************************)

PROCEDURE DISPLAYSPECS;

    VAR
        I,                                 (* Iteration counter      *)
        WIDTH:                             (* Number of fields       *)
               INTEGER;

    BEGIN
        WIDTH:=SPECS2[0];
        GOTOXY(0,7);
        FOR I:=1 TO WIDTH DO
            WRITELN(I:7,SPECS2[I]:12,' ':7,SPECS1[I],CHR(29));
    END;  (* End of DISPLAY SPECS *)

(*******************************************************************)

PROCEDURE GETFLDWIDTH;

    BEGIN
    (*$I-*)
        GOTOXY(60,5);
        WRITE(CHR(15),COLS:3,CHR(14));
        GOTOXY(76,5);
        WRITE(CHR(15),(80-COLS):3,CHR(14));
        GOTOXY(16,ROW);
        WRITELN(CHR(29));
        GOTOXY(18,ROW);

        RESET(INPUT);
        READ(SPECS2[INDEX]);

        WHILE (IORESULT=14) OR (SPECS2[INDEX]<8) OR
              (SPECS2[INDEX]>15) OR (COLS+SPECS2[INDEX]>80) DO
            BEGIN
                GOTOXY(16,ROW);
                WRITELN(CHR(7),CHR(29));

                GOTOXY(56,ROW);                    (* Error Messages *)
                IF (SPECS2[INDEX]<8) THEN
                    WRITE(CHR(15),'Must be at least 8',CHR(14));

                IF (SPECS2[INDEX]>15) OR (COLS+SPECS2[INDEX]>80) THEN
                    BEGIN
                        IF (SPECS2[INDEX]>15) AND
                                         (COLS+SPECS2[INDEX]>80) THEN
                            BEGIN
                                IF (80-COLS<15) THEN
                                    WRITE(CHR(15),'Must be no more ',
                                         'than ',80-COLS,CHR(14))
                                ELSE
                                    WRITE(CHR(15),'Must be no more ',
                                         'than 15',CHR(14))
                            END
                        ELSE
                            BEGIN
                                IF (SPECS2[INDEX]>15) THEN
                                    WRITE(CHR(15),'Must be no more ',
                                         'than 15',CHR(14))
                                ELSE
                                    WRITE(CHR(15),'Must be no more ',
                                         'than ',80-COLS,CHR(14))
                            END
                    END; (* End of Error Messages *)

                GOTOXY(18,ROW);
                RESET(INPUT);
                READ(SPECS2[INDEX]);
            END;  (* End of Bad Width *)

        COLS:=COLS+SPECS2[INDEX];
        GOTOXY(60,5);
        WRITE(CHR(15),COLS:3,CHR(14));
        GOTOXY(76,5);
        WRITE(CHR(15),(80-COLS):3,CHR(14));
    (*$I+*)
    END;  (* End of GET FIELD WIDTH *)

(*********************************************************************)

PROCEDURE GETFLDNAME;

    BEGIN
    (*$I-*)
        GOTOXY(26,ROW);
        WRITE(CHR(29));
        RESET(INPUT);
        READLN(SPECS1[INDEX]);

        IF (LENGTH(SPECS1[INDEX])>SPECS2[INDEX]) THEN
            WHILE (LENGTH(SPECS1[INDEX])>SPECS2[INDEX]) AND
                  (POS(' ',SPECS1[INDEX])=1) DO
                      DELETE(SPECS1[INDEX],1,1);

        IF (LENGTH(SPECS1[INDEX])>SPECS2[INDEX]) THEN
            SPECS1[INDEX]:=COPY(SPECS1[INDEX],1,SPECS2[INDEX]);
    (*$I++)
    END;  (* End of GET FIELD NAME *)

(*******************************************************************)

PROCEDURE ADDAFIELD;

(*******************************************************************)
(*                                                                 *)
(*        This procedure adds a field or variable to a data        *)
(*             file, if there is room, which is then filled        *)
(*             with either computed values or user input           *)
(*             values.  The Number of Fields counter, WIDTH,       *)
(*             which is stored in SPECS2[0], is updated.           *)
(*                                                                 *)
(*******************************************************************)

    VAR
        COLS,                          (* Number of columns in record *)
        I,                             (* Iteration counter           *)
        INDEX,                         (* Field added to data array   *)
        NUMREC,                        (* Number of records           *)
        ROOM,                          (* Number of free fields       *)
        ROW,                           (* Row on screen               *)
        WIDTH:                         (* Number of fields            *)
               INTEGER;
        OPT1,                          (* Menu options                *)
        OPT2:
               CHAR;
        DONE:                          (* Completion indicator        *)
               BOOLEAN;

(*******************************************************************)
(*                       Internal Procedures                       *)
(*******************************************************************)

    PROCEDURE DEFINEFIELD;

        BEGIN
            WIDTH:=SPECS2[0];
            COLS:=0;
            FOR I:=1 TO WIDTH DO
                COLS:=COLS+SPECS2[I];

            GOTOXY(0,5);
            WRITELN(CHR(15),'FIELD NUMBER','WIDTH':9,'NAME':8,
                    CHR(14));

            GOTOXY(50,5);
            WRITELN(CHR(15),'COLS USED:',COLS:3,'    COLS LEFT:',
                   (80-COLS):3,CHR(14));
            DISPLAYSPECS(SPECS1,SPECS2);

            GOTOXY(0,19);
            WRITELN('Now enter width and name for the ',
                   'added field.  The width must be at ');
            WRITELN('least 8 and no more than 15.  This ',
                   'includes room for 6 significant digits.');
            WRITELN('The name should be no wider than ',
                   'the field.  Finally, remember the');
            WRITELN('upper limit of 80 characters ',
                   'per record.');

            INDEX:=WIDTH+1;                      (* New field specs *)
            ROW:=INDEX+6;
            GOTOXY(0,ROW);
            WRITE(INDEX:7);

            GETFLDWIDTH(SPECS2,INDEX,ROW,COLS);

            GETFLDNAME(SPECS1,SPECS2,INDEX,ROW);

            ERASE(19,4);
            GOTOXY(0,19);
            WRITELN('Select desired option:');
            WRITELN('        1 - Change specifications');
            WRITELN('        2 - Proceed with ADD');
            WRITELN;
            GETOPTION(OPT1);
            WHILE (OPT1<>'1') AND (OPT1<>'2') DO
                GETOPTION(OPT1);
            ERASE(19,3);
        END;  (* End of DEFINE the FIELD *)

(****************************************************************)
(*                   Main body of ADDAFIELD                     *)
(****************************************************************)

    BEGIN
        NUMREC:=SPECS2[-1];            (* Initialize parameters      *)
        WIDTH:=SPECS2[0];
        ROOM:=MAXSIZE-WIDTH;
        DONE:=FALSE;

        WRITELN(CHR(12),' ':30,CHR(15),' ADD FIELD ROUTINE ',
                CHR(14));
        GOTOXY(0,5);

        IF (ROOM=0) THEN               (* No room to add a field     *)
            BEGIN
                GOTOXY(1,21);
                WRITELN(CHR(15),'WARNING:',CHR(14),'  The file ',
                       'is full.  No more fields can be added.');
                WRITELN(' ':11,'Press any key to continue');
                GOTOXY(0,21);
                GETOPTION(OPT1);
            END
        ELSE
            BEGIN                      (* Room available to add field *)
```

```
                WRITELN('Select desired option:');
                WRITELN('        1 - Proceed with ADD');
                WRITELN('        2 - Exit ADD FIELD');
                GETOPTION(OPT1);
                WHILE (OPT1<>'1') AND (OPT1<>'2') DO
                    GETOPTION(OPT1);
                ERASE(5,3);

                IF (OPT1='2') THEN
                    DONE:=TRUE;

                WHILE (OPT1='1') DO
                    DEFINEFIELD;

                IF NOT(DONE) THEN                   (* Fill the field *)
                    BEGIN
                        SPECS2[0]:=INDEX;           (* New WIDTH      *)
                        FOR I:=1 TO NUMREC DO
                            DATA[I,INDEX]:=0.0;
                        FILLFIELD(DATA,SPECS1,SPECS2,INDEX);
                    END;

            END;  (* End of Room Available *)

    END;  (* End of ADD A FIELD *)

(*********************************************************************)

PROCEDURE CHGAFIELD;

(*********************************************************************)
(*                                                                   *)
(*          This procedure changes both the specifications for       *)
(*               and contents of one data field.  A check is         *)
(*               made to keep each record under 81 columns.          *)
(*                                                                   *)
(*********************************************************************)

    VAR
        COLS,                           (* Number of columns per record *)
        I,                              (* Iteration counter            *)
        INDEX,                          (* Field to be changed          *)
        NUMREC,                         (* Number of records in file    *)
        ROW,                            (* Row on screen                *)
        WIDTH:                          (* Number of fields in file     *)
               INTEGER;
        OPT1,
        OPT2,                           (* Menu options                 *)
        OPT3:
               CHAR;
        DONE:                           (* Completion indicator         *)
               BOOLEAN;

(*********************************************************************)
(*                      Internal procedures                          *)
(*********************************************************************)

    PROCEDURE PICKOPTION;

        BEGIN
        WRITELN(CHR(12),' ':25,CHR(15),' CHANGE FIELD ROUTINE ',
                CHR(14));
        GOTOXY(0,5);
        WRITELN('This routine will allow you to change the ',
                'contents of a currently defined and');
        WRITELN('full field in the data array.  Once started ',
                'that field may become contaminated.');
        GOTOXY(0,10);
        WRITELN('Select desired option:');
        WRITELN('        1 - Proceed with CHANGE');
        WRITELN('        2 - Exit CHANGE FIELD');
    END;

(*******************************************************************)

   PROCEDURE CHECKSPECS;

    BEGIN
        WRITELN(CHR(12),' ':25,CHR(15),' CURRENT FIELD ',
                'SPECIFICATIONS ',CHR(14));
        GOTOXY(0,5);
        WRITELN(CHR(15),'FIELD NUMBER','WIDTH':9,
                'NAME':8,CHR(14));

        GOTOXY(50,5);
        WRITELN(CHR(15),'COLS USED:',COLS:3,'    COLS LEFT:',
                (80-COLS):3,CHR(14));
        DISPLAYSPECS(SPECS1,SPECS2);

        GOTOXY(0,19);
        WRITELN('Select desired option:');
        WRITELN('        1 - Proceed with CHANGE');
        WRITELN('        2 - Exit CHANGE FIELD');
    END;  (* End of CHECK SPECificationS *)

(*******************************************************************)

    PROCEDURE MAKECHANGE;

    BEGIN
        ERASE(18,5);
        GOTOXY(0,20);
        WRITELN('Now enter new WIDTH and NAME for FIELD # ',
                 INDEX);
        ROW:=18;
        COLS:=COLS-SPECS2[INDEX];

        GETFLDWIDTH(SPECS2,INDEX,ROW,COLS);

        GETFLDNAME(SPECS1,SPECS2,INDEX,ROW);

        DISPLAYSPECS(SPECS1,SPECS2);

        ERASE(18,5);
        GOTOXY(0,19);
        WRITELN('Select desired option:');
        WRITELN('        1 - Change specifications of field # ',
                    INDEX,'.');
        WRITELN('        2 - Proceed with CHANGE FIELD');
        GETOPTION(OPT3);
```

```
          WHILE (OPT3<>'1') AND (OPT3<>'2') DO
              GETOPTION(OPT3);

          ERASE(19,3);
          IF (OPT3='2') THEN
              DONE:=TRUE;
      END;  (* End of MAKE CHANGE *)

(******************************************************************)
(*                        Main body of CHGAFIELD                   *)
(******************************************************************)

  BEGIN
  (*$I-*)
      NUMREC:=SPECS2[-1];             (* Initialize parameters    *)
      WIDTH:=SPECS2[0];
      DONE:=FALSE;
      COLS:=0;

      PICKOPTION;

      GETOPTION(OPT1);
      WHILE (OPT1<>'1') AND (OPT1<>'2') DO
          GETOPTION(OPT1);
      ERASE(5,8);

      IF (OPT1='1') THEN              (* Do a Change              *)
          BEGIN
              FOR I:=1 TO WIDTH DO
                  COLS:=COLS+SPECS2[I];
              CHECKSPECS;

              GETOPTION(OPT2);
              WHILE (OPT2<>'1') AND (OPT2<>'2') DO
                  GETOPTION(OPT2);
              ERASE(18,5);

              IF (OPT2='1') THEN      (* Accomplish change        *)
                  BEGIN
                      GOTOXY(0,20);
                      WRITELN('Enter field to be changed: ',
                              '(1 to ',WIDTH,')');
                      WRITELN('       (0 = Skip change)');
                      GOTOXY(6,18);
                      RESET(INPUT);
                      READ(INDEX);

                      WHILE (IORESULT=14) OR (INDEX<1) OR
                                           (INDEX>WIDTH) DO
                          BEGIN
                              GOTOXY(1,22);
                              WRITE(CHR(15),'WARNING:',CHR(14),
                                    '  Bad field number.  ',
                                    'Press any key to try ',
                                    'again.');
                              GOTOXY(0,22);
                              GETOPTION(OPT3);
                              ERASE(22,1);
                              ERASE(18,2);
```

```
                    GOTOXY(6,18);
                    RESET(INPUT);
                    READ(INDEX);
                END;  (* End of Bad Index *)

            WHILE NOT(DONE) DO
                MAKECHANGE;

            FILLFIELD(DATA,SPECS1,SPECS2,INDEX);

        END;  (* End of Accomplish Change *)
    END;  (* End of Do a Change *)
(*$I+*)
END;  (* End of CHG A FIELD *)

(**************************************************************)

PROCEDURE FILLFIELD;

(**************************************************************)
(*                                                            *)
(*      This procedure displays the menu of the options       *)
(*          available  for filling the specified field        *)
(*          in procedures ADDAFIELD and CHGAFIELD.  It        *)
(*          then calls the appropriate procedure.             *)
(*                                                            *)
(**************************************************************)

    VAR
        I,                              (* Iteration counter    *)
        NUMREC:                         (* Number of records    *)
                INTEGER;
        OPT:                            (* Menu option          *)
                CHAR;
        DONE:                           (* Completion indicator *)
                BOOLEAN;

(**************************************************************)
(*                    Internal Procedure                      *)
(**************************************************************)

    PROCEDURE DEFINEOPTS;

        VAR OPT: CHAR;              (* End of display indicator   *)

        BEGIN
            WRITELN(CHR(12),' ':26,CHR(15),' FILL FIELD OPTIONS ',
                    CHR(14));
            GOTOXY(0,5);
            WRITELN('Option');
            WRITELN('------');
            WRITELN('   1 - Fills specified field with user ',
                    'selected constants; based');
            WRITELN('       on partition(s) within that or a ',
                    'different field.',CHR(13));
            WRITELN('   2 - Computes and stores in the specified ',
                    'field the results of one or more');
            WRITELN('       arithmetic operations on one or more ',
                    'fields.',CHR(13));
            WRITELN('   3 - Accepts data as input by the user at ',
                'the keyboard; one record at a time.',CHR(13));
          WRITELN('   4 - Display these definitions',CHR(13));
          WRITELN('   5 - Exit FILL FIELD routine');
          GOTOXY(22,22);
          WRITE('Press any key to continue   ');
          GETOPTION(OPT);
      END;  (* End of DEFINE OPTS *)

(****************************************************************)
(*                     Main body of FILLFIELD                  *)
(****************************************************************)

    BEGIN
    (*$R MU_B *)                          (* Retain UNIT in memory *)
      DONE:=FALSE;
      NUMREC:=SPECS2[-1];

      WHILE NOT(DONE) DO
          BEGIN
              WRITELN(CHR(12),' ':26,CHR(15),' FILL FIELD ',
                      'ROUTINE ',CHR(14));
              GOTOXY(0,5);
              WRITELN('Select desired option:',CHR(13));
              WRITELN('        1 - Recode');
              WRITELN('        2 - Compute');
              WRITELN('        3 - User input');
              WRITELN('        4 - Define above options');
              WRITELN('        5 - Exit FILL FIELD');
              GETOPTION(OPT);
              WHILE (OPT<'1') AND (OPT>'5') DO
                  GETOPTION(OPT);

              CASE (OPT) OF
                '1': RECODE(DATA,SPECS1,SPECS2,INDEX);
                '2': COMPUTE(DATA,SPECS1,SPECS2,INDEX);
                '3': USERINPUT(DATA,SPECS1,SPECS2,INDEX);
                '4': DEFINEOPTS;
                '5': FOR I:=1 TO NUMREC DO
                         IF (DATA[I,INDEX]<>0.0) THEN
                             DONE:=TRUE;
              END;  (* End of CASE *)

              IF (OPT='5') AND NOT(DONE) THEN
                  BEGIN
                      GOTOXY(0,18);
                      WRITELN(CHR(15),' WARNING:',CHR(14),
                              '  Field is currently all zero''s');
                      WRITELN;
                      WRITELN(' ':11,'Select desired option:');
                      WRITELN(' ':11,'        1 - Go back and ',
                              'fill field');
                      WRITE(' ':11,'        2 - Leave field ',
                            'all zero''s   ');
                      GETOPTION(OPT);
                      WHILE (OPT<>'1') AND (OPT<>'2') DO
                          GETOPTION(OPT);

                      IF (OPT='2') THEN
                          DONE:=TRUE;
                  END;  (* End of error exit attempt *)
              END;  (* End of WHILE loop *)
      END;  (* End of FILLFIELD *)

(*****************************************************************)

(*$I PSPP:MODIFILE *)                    (* Include procedure in UNIT *)

(*****************************************************************)

(*$I PSPP:USERINPUT *)                   (* Include procedure in UNIT *)

(*****************************************************************)
(*                    Initialization part of UNIT                  *)
(*****************************************************************)

END.
```

```
PROCEDURE MODIFILE;

(****************************************************************)
(*                                                              *)
(*      This procedure needs as input:                          *)
(*                                                              *)
(*            DATA - Array of raw data to be modified           *)
(*            SPECS1 - Array of field or variable names         *)
(*            SPECS2 - Array of field widths                    *)
(*                                                              *)
(*      This procedure returns as output the above arrays       *)
(*            after modification.  Changes include addition,    *)
(*            removal, and modification of both records and     *)
(*            fields.                                           *)
(*                                                              *)
(****************************************************************)

    VAR
        DONE:                               (* Completion indicator *)
            BOOLEAN;
        OPT:                                (* Menu option          *)
            CHAR;

(****************************************************************)
(*                      Internal Procedures                     *)
(****************************************************************)

    PROCEDURE DISPLAYWARNING;

        BEGIN
            WRITELN(CHR(12),' ':25,CHR(15),' MODIFY DATA ROUTINE ',
                    CHR(14));
            GOTOXY(0,5);
            WRITELN(CHR(15),' WARNING:',CHR(14),'  You should ',
                   'save all data changes as soon as');
            WRITELN('             possible or risk losing them.');
            GOTOXY(0,10);
            WRITELN('    NOTE:  If you save the modified data ',
                    'using the same name as before,');
            WRITELN('             you will overwrite the ',
                    'unmodified data.');
            GOTOXY(22,22);
            WRITE('Press any key to continue   ');
            GETOPTION(OPT);
        END;  (* End of DISPLAYWARNING *)

(****************************************************************)

    PROCEDURE DISPLAYMENU;

        BEGIN
            WRITELN(CHR(12),' ':25,CHR(15),' MODIFY DATA ROUTINE ',
                    CHR(14));
            GOTOXY(0,5);
            WRITELN('Select desired option:');
            WRITELN('        1 - Add a record');
            WRITELN('        2 - Delete a record');
            WRITELN('        3 - Add a field');
            WRITELN('        4 - Delete a field');
            WRITELN('        5 - Change a record');
            WRITELN('        6 - Change a field');
            WRITELN('        7 - Exit MODIFY DATA routine');
            GETOPTION(OPT);
            WHILE (OPT<'1') OR (OPT>'7') DO
                GETOPTION(OPT);

            CASE (OPT) OF
               '1': ADDARECORD(DATA,SPECS1,SPECS2);
               '2': SUBARECORD(DATA,SPECS1,SPECS2);
               '3': ADDAFIELD(DATA,SPECS1,SPECS2);
               '4': SUBAFIELD(DATA,SPECS1,SPECS2);
               '5': CHGARECORD(DATA,SPECS1,SPECS2);
               '6': CHGAFIELD(DATA,SPECS1,SPECS2);
               '7': DONE:=TRUE;
            END;  (* End of CASE *)
        END;  (* End of DISPLAY MENU *)

(*******************************************************************)
(*                     Main body of MODIFILE                       *)
(*******************************************************************)

    BEGIN
        DONE:=FALSE;
        WHILE NOT(DONE) DO
            DISPLAYMENU;
        DISPLAYWARNING;
    END;  (* End of MODIfy FILE *)

(*******************************************************************)
```

```
PROCEDURE USERINPUT;

(*****************************************************************)
(*                                                               *)
(*        This procedure is used to fill a field of the          *)
(*             data array; one record at a time.                 *)
(*                                                               *)
(*****************************************************************)

    VAR
        FIELDWIDTH,                   (* Max field width of value  *)
        I,                            (* Iteration counter         *)
        NUMREC:                       (* Number of records in file *)
                   INTEGER;
        VALUE:                        (* Value as input by user    *)
                   REAL;
        OPT1,                         (* Menu options              *)
        OPT2:
                   CHAR;
        NAME:                         (* Field or variable name    *)
                   STRING;

(*****************************************************************)
(*                    Internal Procedures                        *)
(*****************************************************************)

    PROCEDURE DISPLAYHEADING;

        BEGIN
            ERASE(5,8);
            GOTOXY(0,5);
            WRITE(CHR(15),' FIELD NAME:',CHR(14),NAME:15);
            GOTOXY(38,5);
            WRITE(CHR(15),' MAX WIDTH:',CHR(14));
            GOTOXY(50,5);
            FOR I:=1 TO FIELDWIDTH DO
                WRITE(CHR(15),'X',CHR(14));
            WRITELN('  (',FIELDWIDTH,')');
        END;

(*****************************************************************)

    PROCEDURE TAKEINPUTS;

        BEGIN
        (*$I-*)
            GOTOXY(4,8);
            WRITE(CHR(15),' RECORD:',CHR(14),'  #  ',I:3,
                 ' of ',NUMREC);
            GOTOXY(50,8);
            RESET(INPUT);
            READ(VALUE);

            WHILE (IORESULT=14) DO                  (* Bad Value *)
                BEGIN
                    WRITE(CHR(8),CHR(29));
                    GOTOXY(1,20);
                    WRITE(CHR(15),'WARNING:',CHR(14),'  Value must ',
                         'be a number.  Press any key to continue.');
                    GOTOXY(0,20);
                    GETOPTION(OPT2);
                    ERASE(20,1);
                    GOTOXY(50,8);
                    RESET(INPUT);
                    READ(VALUE);
                END;   (* End of Bad Value *)

            DATA[I,INDEX]:=VALUE;
            ERASE(8,1);
        (*$I+*)
        END;   (* End of TAKE INPUTS *)

(******************************************************************)
(*                     Main body of USERINPUT                      *)
(******************************************************************)

    BEGIN
        NUMREC:=SPECS2[-1];              (* Initialize parameters     *)
        NAME:=SPECS1[INDEX];
        IF (LENGTH(NAME)>15) THEN        (* Truncate to fit           *)
            NAME:=COPY(NAME,1,15);
        FIELDWIDTH:=SPECS2[INDEX];

        WRITELN(CHR(12),' ':26,CHR(15),' USER INPUT ROUTINE ',
                CHR(14));
        GOTOXY(0,5);
        WRITELN('Any values input that exceed the MAX WIDTH ',
                'will mess up the columns in the');
        WRITELN('ECHOFILE routine.  To prevent, run ''Change ',
                'a field'' to expand the field width');
        WRITELN('after entering all records, then exit ',
                'FILLFIELD without changing field contents.');

        GOTOXY(0,10);
        WRITELN('Select desired option:');
        WRITELN('        1 - Proceed with INPUT');
        WRITELN('        2 - Exit USER INPUT');
        GETOPTION(OPT1);
        WHILE (OPT1<>'1') AND (OPT1<>'2') DO
            GETOPTION(OPT1);

        IF (OPT1='1') THEN                    (* Start inputting      *)
            BEGIN
                DISPLAYHEADING;

                FOR I:=1 TO NUMREC DO
                    TAKEINPUTS;
            END;  (* End of Start Inputting *)

    END;     (* End of USER INPUT *)

(******************************************************************)
```

```
(*$S+*)

UNIT MU_E; INTRINSIC CODE 15;

INTERFACE
    USES MAIN_UNIT;

    PROCEDURE LOADDATA(VAR DATA:RAWDATA;VAR SPECS1:HEADER1;
                       VAR SPECS2:HEADER2);

    PROCEDURE SAVEFILE(VAR DATA:RAWDATA;VAR SPECS1:HEADER1;
                       VAR SPECS2:HEADER2);

    PROCEDURE ECHOFILE(VAR DATA:RAWDATA;VAR SPECS1:HEADER1;
                       VAR SPECS2:HEADER2;PRINTER:BOOLEAN);

IMPLEMENTATION

(*********************************************************************)
(*                        Main part of MU_E                          *)
(*********************************************************************)

PROCEDURE LOADDATA;

(*********************************************************************)
(*                                                                   *)
(*          This procedure requires existence of data files          *)
(*               saved via the SAVEFILE procedure                    *)
(*                                                                   *)
(*          This procedure returns as output:                        *)
(*                                                                   *)
(*               DATA - Array of raw data as stored on disk          *)
(*               SPECS1 - Array of field names                       *)
(*               SPECS2 - Array of field widths                      *)
(*                                                                   *)
(*********************************************************************)

    VAR
        I,J,                             (* Indexes                  *)
        NUMREC,                          (* Number of records in file *)
        WIDTH:                           (* Number of fields in file  *)
                   INTEGER;
        FILEID,                          (* File name as input by user *)
        FILENAME:                        (* File name as stored       *)
                   STRING[21];
        OPT1,                            (* Menu options              *)
        OPT2:
                   CHAR;

(*********************************************************************)
(*                      Internal Procedures                          *)
(*********************************************************************)

    PROCEDURE FILEFOUND;

        BEGIN
            GOTOXY(0,15);
            WRITELN('Loading ',FILEID,'. . .Please stand by');

            READLN(DATAFILE,SPECS2[-1],SPECS2[0]);
            NUMREC:=SPECS2[-1];
            WIDTH:=SPECS2[0];

            FOR I:=0 TO WIDTH DO
                READLN(DATAFILE,SPECS1[I]);
            FOR I:=1 TO WIDTH DO
                READLN(DATAFILE,SPECS2[I]);
            FOR I:=1 TO NUMREC DO
                FOR J:=1 TO WIDTH DO
                    READLN(DATAFILE,DATA[I,J]);

            GOTOXY(0,17);
            WRITE('Load complete. Press any key to continue   ');
            GETOPTION(OPT2);
            WRITELN(CHR(12));                   (* Clear screen      *)
            OPT1:='2';
        END;  (* End of FILE FOUND *)

(*******************************************************************)

    PROCEDURE FILENOTFOUND;

        BEGIN
            GOTOXY(0,15);
            WRITELN('Specified file not found',CHR(13));
            WRITELN('Select desired option:');
            WRITELN('       1 - Try another load');
            WRITELN('       2 - Exit LOAD Procedure');
            GETOPTION(OPT1);
            WHILE (OPT1<>'1') AND (OPT1<>'2') DO
                GETOPTION(OPT1);
            ERASE(12,8);
            GOTOXY(0,12);
        END;  (* End of FILE NOT FOUND *)

(*******************************************************************)
(*                      Main body of LOAD DATA                     *)
(*******************************************************************)

    BEGIN
    (*$I-*)
        WRITELN(CHR(12),' ':26,CHR(15),' LOAD DATA ROUTINE ',
                CHR(14));
        GOTOXY(0,5);
        WRITELN('Select desired option:');
        WRITELN('       1 - Load a data file');
        WRITELN('       2 - Exit LOAD routine');
        GETOPTION(OPT1);
        WHILE (OPT1<>'1') AND (OPT1<>'2') DO
            GETOPTION(OPT1);

        WHILE (OPT1='1') DO           (* Attempt a load              *)
            BEGIN
                GOTOXY(0,10);
                WRITELN('Enter desired file name: ',
                        '(1 to 10 characters)');
                GOTOXY(0,12);
                RESET(INPUT);
                READLN(FILEID);
                IF (LENGTH(FILEID)>10) THEN
                    FILEID:=COPY(FILEID,1,10);
                FILENAME:=CONCAT('BLANK:',FILEID,'.TEXT');

                RESET(DATAFILE,FILENAME);

                IF (IORESULT=0) THEN      (* File found                    *)
                    FILEFOUND
                ELSE
                    FILENOTFOUND;

                CLOSE(DATAFILE);
            END;  (* End of Attempt load *)
    (*$I+*)
    END;  (* End of LOADDATA *)

(*********************************************************************)

PROCEDURE SAVEFILE;

(*********************************************************************)
(*                                                                   *)
(*          This procedure needs as input:                           *)
(*                                                                   *)
(*               DATA - Array of data to be saved                    *)
(*               SPECS1 - Array of field names                       *)
(*               SPECS2 - Array of field widths                      *)
(*                                                                   *)
(*          This procedure stores the data file on disk              *)
(*               under the name BLANK:<name>.TEXT                    *)
(*                                                                   *)
(*********************************************************************)

    VAR
        I,J,                            (* Indexes                     *)
        NUMREC,                         (* Number of records in file   *)
        WIDTH:                          (* Number of fields in file    *)
                    INTEGER;
        FILEID,                         (* File name as input by user  *)
        FILENAME:                       (* File name as stored         *)
                    STRING[21];
        OPT1,                           (* Menu options                *)
        OPT2:
                    CHAR;

(*********************************************************************)
(*                         Internal procedure                        *)
(*********************************************************************)

    PROCEDURE BADSAVE;

        BEGIN
            GOTOXY(0,15);
            WRITELN(CHR(7),'Save not possible. Make sure a ',
                    'properly formatted disk with');
            WRITELN('enough space is available in ',
                    'Drive #2 and the filename is 10 or');
            WRITELN('less characters starting with a letter.',
                    CHR(13));
            WRITELN('Select desired option:');
```

```
          WRITELN('        1 - Try another save');
          WRITELN('        2 - Exit SAVE Procedure');
          GETOPTION(OPT1);
          WHILE (OPT1<>'1') AND (OPT1<>'2') DO
              GETOPTION(OPT1);
          ERASE(10,13);
      END;

(*****************************************************************)
(*                      Main body of SAVEFILE                    *)
(*****************************************************************)

   BEGIN
   (*$I-*)
       NUMREC:=SPECS2[-1];
       WIDTH:=SPECS2[0];

       WRITELN(CHR(12),' ':24,CHR(15),' SAVE DATA FILE ROUTINE ',
               CHR(14));
       GOTOXY(0,5);
       WRITELN('Select desired option:');
       WRITELN('        1 - Save a data file');
       WRITELN('        2 - Exit SAVE routine');
       GETOPTION(OPT1);
       WHILE (OPT1<>'1') AND (OPT1<>'2') DO
           GETOPTION(OPT1);

       WHILE (OPT1='1') DO            (* Attempt a save              *)
           BEGIN
               GOTOXY(0,10);
               WRITELN('Enter desired file name:  ',
                       '(1 to 10 characters)');
               GOTOXY(0,12);
               RESET(INPUT);
               READLN(FILEID);

               IF (LENGTH(FILEID)>10) THEN
                   FILEID:=COPY(FILEID,1,10);
               FILENAME:=CONCAT('BLANK:',FILEID,'.TEXT');

               REWRITE(DATAFILE,FILENAME);

               IF (IORESULT=0) THEN  (* File properly opened      *)
                   BEGIN
                       GOTOXY(0,15);
                       WRITELN('Saving ',FILEID,'. . .',
                               'Please stand by');
                       SPECS1[0]:=FILEID;

                       WRITELN(DATAFILE,SPECS2[-1],' ',SPECS2[0]);
                       FOR I:=0 TO WIDTH DO
                           WRITELN(DATAFILE,SPECS1[I]);
                       FOR I:=1 TO WIDTH DO
                           WRITELN(DATAFILE,SPECS2[I]);
                       FOR I:=1 TO NUMREC DO
                           FOR J:=1 TO WIDTH DO
                               WRITELN(DATAFILE,DATA[I,J]);

                       IF (IORESULT=0) THEN
                           BEGIN
```

```
                                        GOTOXY(0,17);
                                        WRITE('Save complete.  Press any ',
                                              'key to continue   ');
                                        GETOPTION(OPT2);
                                        WRITELN(CHR(12));
                                        OPT1:='2';
                                    END
                                ELSE
                                    BADSAVE;
                            END
                        ELSE
                            BADSAVE;

                    END;  (* End of Attempt Save *)
                CLOSE(DATAFILE,LOCK);
            (*$I+*)
            END;  (* End of SAVEFILE *)

(******************************************************************)

PROCEDURE ECHOFILE;

(******************************************************************)
(*                                                                *)
(*        This procedure needs as input:                          *)
(*                                                                *)
(*             DATA - Array of data to be printed                 *)
(*             SPECS1 - Array of field names                      *)
(*             SPECS2 - Array of field widths                     *)
(*             PRINTER - Set if printer available                 *)
(*                                                                *)
(*        This procedure provides an echocheck of data to         *)
(*             screen and printer (if desired)                    *)
(*                                                                *)
(******************************************************************)

    VAR
        I,                      (* Iteration counter              *)
        FIELDWIDTH,             (* Augmented width used by printer *)
        INDEX,                  (* Index into various arrays      *)
        NUMREC,                 (* Number of records in file      *)
        WIDTH:                  (* Number of fields in file       *)
                 INTEGER;
        FIELDS:                 (* Data fields to be echoed       *)
                 VECTOR;
        OPT1,                   (* Various menu options           *)
        OPT2,
        OPT3:
                 CHAR;

(******************************************************************)
(*                        Internal  Procedures                    *)
(******************************************************************)

    PROCEDURE DISPLAYFIELDS;

        BEGIN
            GOTOXY(0,7);

            FOR I:=1 TO WIDTH DO        (* Display selected fields  *)
```

```
                                GOTOXY(0,17);
                                WRITE('Save complete.  Press any ',
                                      'key to continue   ');
                                GETOPTION(OPT2);
                                WRITELN(CHR(12));
                                OPT1:='2';
                            END
                        ELSE
                            BADSAVE;
                    END
                ELSE
                    BADSAVE;

            END;  (* End of Attempt Save *)
        CLOSE(DATAFILE,LOCK);
    (*$I+*)
    END;  (* End of SAVEFILE *)

(*******************************************************************)

PROCEDURE ECHOFILE;

(*******************************************************************)
(*                                                                 *)
(*        This procedure needs as input:                           *)
(*                                                                 *)
(*             DATA - Array of data to be printed                  *)
(*             SPECS1 - Array of field names                       *)
(*             SPECS2 - Array of field widths                      *)
(*             PRINTER - Set if printer available                  *)
(*                                                                 *)
(*        This procedure provides an echocheck of data to          *)
(*             screen and printer (if desired)                     *)
(*                                                                 *)
(*******************************************************************)

    VAR
        I,                    (* Iteration counter                 *)
        FIELDWIDTH,           (* Augmented width used by printer   *)
        INDEX,                (* Index into various arrays         *)
        NUMREC,               (* Number of records in file         *)
        WIDTH:                (* Number of fields in file          *)
                 INTEGER;
        FIELDS:               (* Data fields to be echoed          *)
                 VECTOR;
        OPT1,                 (* Various menu options              *)
        OPT2,
        OPT3:
                 CHAR;

(*******************************************************************)
(*                       Internal  Procedures                      *)
(*******************************************************************)

    PROCEDURE DISPLAYFIELDS;

        BEGIN
            GOTOXY(0,7);

            FOR I:=1 TO WIDTH DO       (* Display selected fields  *)
```

```
                IF (FIELDS[I]=1) THEN      (* Set to 1 if selected  *)
                    BEGIN
                        WRITE(CHR(15),I:2,'.  ',SPECS1[I]);
                        GOTOXY(20,I+6);
                        WRITELN('ON ',CHR(14));
                    END
                ELSE
                    BEGIN
                        WRITE(I:2,'.  ',SPECS1[I]);
                        GOTOXY(20,I+6);
                        WRITELN('OFF');
                    END;

        END;  (* End of DISPLAY FIELDS *)

(*****************************************************************)

    PROCEDURE CHANGEFIELDS;

        VAR OPT:  CHAR;                      (* Menu Option          *)

        BEGIN
        (*$I-*)
            GOTOXY(0,19);
            WRITELN('Any changes to above list?',CHR(13));
            WRITELN('    1 - Go with list as is');
            WRITELN('    2 - Change(s) required');
            GETOPTION(OPT);
            WHILE (OPT<>'1') AND (OPT<>'2') DO
                GETOPTION(OPT);

            WHILE (OPT='2') DO               (* Make changes         *)
                BEGIN
                    ERASE(19,4);
                    GOTOXY(0,19);
                    WRITELN('Enter field number to change',CHR(13));
                    RESET(INPUT);
                    READ(INDEX);

                    IF (IORESULT=14) OR (INDEX<1) OR
                                        (INDEX>WIDTH) THEN
                        BEGIN
                            GOTOXY(1,22);
                            WRITELN(CHR(15),'WARNING:',CHR(14),
                                   '  Bad field designator. Press ',
                                   'any key to try again.');
                            GOTOXY(0,22);
                        END
                    ELSE
                        BEGIN
                            IF (FIELDS[INDEX]=1) THEN
                                FIELDS[INDEX]:=0
                            ELSE
                                FIELDS[INDEX]:=1;
                            WRITELN('Field ',CHR(15),SPECS1[INDEX],
                                    CHR(14),' changed. Press any ',
                                    'key to continue.  ');
                        END;

                    GETOPTION(OPT2);
                    ERASE(19,4);
                    DISPLAYFIELDS;

                    GOTOXY(0,19);
                    WRITELN('Any changes to above list?');
                    WRITELN('    1 - Go with list as is');
                    WRITELN('    2 - Change(s) required');
                    GETOPTION(OPT);
                    WHILE (OPT<>'1') AND (OPT<>'2') DO
                        GETOPTION(OPT);
                END;  (* End of WHILE loop *)
        (*$I+*)
        END;  (* End of CHANGE FIELDS *)

(***************************************************************************)

    PROCEDURE LIMITECHO;

        BEGIN
            ERASE(5,6);
            GOTOXY(0,5);
            WRITELN('Select fields to be echoed: ',CHR(15),
                   '(1-Yes,2-No)',CHR(14),CHR(13));

            FOR I:=1 TO WIDTH DO                    (* Get fields    *)
                BEGIN
                    WRITE(I:2,'.  ',SPECS1[I],'   ');
                    GETOPTION(OPT2);
                    WHILE (OPT2<>'1') AND (OPT2<>'2') DO
                        GETOPTION(OPT2);
                    WRITELN;
                    FIELDS[I]:=-(ORD(OPT2)-50);     (* Set to 1 Or 0 *)
                END;  (* End of Get fields *)

            ERASE(5,18);
            GOTOXY(0,5);
            WRITELN('Limited echo check of following fields:');

            DISPLAYFIELDS;
            CHANGEFIELDS;

            ERASE(5,18);
        END;  (* End of LIMIT ECHO *)

(***************************************************************************)

    PROCEDURE ECHODATA;

        BEGIN
            FOR INDEX:=1 TO NUMREC DO               (* Echo the data *)
                BEGIN
                    IF ((INDEX MOD 16)=0) THEN      (* Page pause    *)
                        BEGIN
                            GOTOXY(22,22);
                            WRITE('Press any key to continue   ');
                            GETOPTION(OPT2);
                            ERASE(5,18);
                            GOTOXY(0,5);
                        END;
                    IF (PRINTER) AND (OPT3='2') THEN
                        WRITE(PTR,INDEX:4,' ':4);

                    FOR I:=1 TO WIDTH DO
                        IF (FIELDS[I]=1) THEN
                            BEGIN
                                FIELDWIDTH:=SPECS2[I];
                                WRITE(DATA[INDEX,I]:FIELDWIDTH:5);

                                FIELDWIDTH:=FIELDWIDTH+4;
                                IF (PRINTER) AND (OPT3='2') THEN
                                    WRITE(PTR,
                                        DATA[INDEX,I]:FIELDWIDTH:5);
                            END;
                    WRITELN;
                    IF (PRINTER) AND (OPT3='2') THEN
                        WRITELN(PTR);
                END;
            END;   (* End of ECHO DATA *)

(*****************************************************************)
(*                   Main body of ECHOFILE                       *)
(*****************************************************************)

    BEGIN
        NUMREC:=SPECS2[-1];
        WIDTH:=SPECS2[0];
        WRITELN(CHR(12),' ':26,CHR(15),' ECHO DATA ROUTINE ',
                CHR(14));
        GOTOXY(0,5);
        WRITELN('Do you desire a limited echo check?',CHR(13));
        WRITELN('Select desired option:');
        WRITELN('       1 - Limited echo check');
        WRITELN('       2 - Complete echo check');
        GETOPTION(OPT1);
        WHILE (OPT1<>'1') AND (OPT1<>'2') DO
            GETOPTION(OPT1);
        ERASE(5,5);

        IF (OPT1='1') AND (WIDTH>0) THEN    (* Limited echo check    *)
            LIMITECHO;

        IF (OPT1='2') AND (WIDTH>0) THEN    (* Complete echo check   *)
            FOR I:=1 TO WIDTH DO
                FIELDS[I]:=1;

        IF (PRINTER) THEN                   (* Hardcopy desired?     *)
            BEGIN
                GOTOXY(0,5);
                WRITELN('Do you want a hardcopy?',CHR(13));
                WRITELN('Select desired option:');
                WRITELN('       1 - Screen only');
                WRITELN('       2 - Screen and printer');
                GETOPTION(OPT3);
                WHILE (OPT3<>'1') AND (OPT3<>'2') DO
                    GETOPTION(OPT3);

                IF (OPT3='2') THEN
                    WRITELN(PTR,CHR(15));   (* Compressed printing   *)
                ERASE(5,5);
                END;

        (*  Print the echo check   *)

            IF (PRINTER) AND (OPT3='2') THEN
                BEGIN
                    WRITELN(PTR,'ECHOCHECK OF CURRENT DATAFILE:');
                    WRITE(PTR,CHR(13),'INDEX',' ':3);
                END;

            GOTOXY(0,3);
            FOR I:=1 TO WIDTH DO                  (* Field names            *)
                IF (FIELDS[I]=1) THEN
                    BEGIN
                        WRITE(SPECS1[I]:SPECS2[I]);
                        FIELDWIDTH:=SPECS2[I]+4;
                        IF (PRINTER) AND (OPT3='2') THEN
                            WRITE(PTR,SPECS1[I]:FIELDWIDTH);
                    END;

            WRITELN(CHR(13));                     (* Carrage returns        *)
            IF (PRINTER) AND (OPT3='2') THEN
                WRITELN(PTR,CHR(13));

            ECHODATA;

            GOTOXY(16,22);
            WRITE('End of Echo Data. Press any key to continue   ');
            GETOPTION(OPT2);

            IF (PRINTER) AND (OPT3='2') THEN
                WRITELN(PTR,CHR(13));
        END;  (* End of ECHOFILE *)

(*******************************************************************)
(*                    Initialization part of UNIT                  *)
(*******************************************************************)

END.
```

```
(*$S+*)

UNIT MU_F; INTRINSIC CODE 16;

INTERFACE
    USES TRANSCEND, MAIN_UNIT;

    PROCEDURE ASSIGNVARIABLES(VAR SPECS1:HEADER1;VAR SPECS2,
                              GROUP:HEADER2;CAT:INTEGER;
                              VAR FEAS:BOOLEAN);

    PROCEDURE CALCULATE(VAR XBAR,SDEV:VECTOR;VAR DATA:RAWDATA;
                        VAR SPECS1:HEADER1;VAR GROUP:HEADER2;
                        NUMREC,WIDTH:INTEGER;PRINTER:BOOLEAN);

IMPLEMENTATION

(*****************************************************************)
(*                      Main part of MU_F                       *)
(*****************************************************************)

PROCEDURE ASSIGNVARIABLES;

(*****************************************************************)
(*                                                               *)
(*       This procedure allows the user to select from the       *)
(*            variables in SPECS1 and assign a GROUP value       *)
(*            to variables desired for analysis based on         *)
(*            CATegory (1=CANCOR,2=FACTOR):                      *)
(*                                                               *)
(*            0 - Not selected                                   *)
(*            1 - Criterion Variable      (CANCOR)               *)
(*            2 - Predictor Variable      (CANCOR)               *)
(*            3 - Manifestation Variable  (FACTOR)               *)
(*                                                               *)
(*****************************************************************)

    VAR
        I,                               (* Iteration counter          *)
        INDEX,                           (* Field index into arrays    *)
        P,                               (* Number of Criterions       *)
        K,                               (* Number of Predictors       *)
        N,                               (* Number of Manifestations   *)
        ROW,                             (* Row on screen              *)
        WIDTH:                           (* Number of fields           *)
                INTEGER;
        OPT:                             (* Menu Options               *)
                CHAR;
        DONE:                            (* Completion indicator       *)
                BOOLEAN;

(*****************************************************************)
(*                      Internal Procedures                      *)
(*****************************************************************)

    PROCEDURE CHECKSIZE;

        BEGIN
            CASE (CAT) OF
                1: IF (WIDTH<4) THEN
                BEGIN
                    DONE:=TRUE;
                    FEAS:=FALSE;

                    GOTOXY(1,20);
                    WRITELN(CHR(15),'WARNING:',CHR(14),
                          '  There must be at least 4 ',
                          'variables in the data base to ',
                          'run CANCOR.',CHR(13));
                    WRITE(' ':11,'Press any key to continue.');
                    GOTOXY(1,20);
                    GETOPTION(OPT);
                    ERASE(20,3);
                END;

            2: IF (WIDTH<2) THEN
                BEGIN
                    DONE:=TRUE;
                    FEAS:=FALSE;

                    GOTOXY(1,20);
                    WRITELN(CHR(15),'WARNING:',CHR(14),
                          '  There must be at least 2 ',
                          'variables in the data base ',
                          'to run FACTOR.',CHR(13));
                    WRITE(' ':11,'Press any key to continue.');
                    GOTOXY(0,20);
                    GETOPTION(OPT);
                    ERASE(20,3);
                END;
        END;  (* End of CASE *)
    END;  (* End of CHECK field list SIZE *)

(*****************************************************************)

  PROCEDURE DISPLAYSPECS;

    BEGIN
        GOTOXY(0,7);
        FOR I:=1 TO WIDTH DO
            WRITELN(I:7,SPECS2[I]:12,' ':6,SPECS1[I],CHR(29));
    END;  (* End of DISPLAY SPECS *)

(*****************************************************************)

  PROCEDURE GETVALIDINDEX;

    BEGIN
    (*$I-*)
        RESET(INPUT);
        READ(INDEX);

        WHILE (IORESULT=14) OR (INDEX<1) OR (INDEX>WIDTH) DO
            BEGIN
                GOTOXY(1,21);
                WRITELN(CHR(15),'WARNING:',CHR(14),'  Bad ',
                       'index.  Must be an integer between ',
                       '1 and ',WIDTH,'.');
                WRITELN(' ':11,'Press any key to try again');
                GOTOXY(0,21);
```

```
                    GETOPTION(OPT);
                    ERASE(20,3);
                    GOTOXY(0,20);
                    RESET(INPUT);
                    READ(INDEX);
                END;
        (*$I+*)
        END;  (* End of GET VALID index *)

(*****************************************************************)

    PROCEDURE ASSIGNCRITERION;

        BEGIN
            WRITELN('Enter index (1-',WIDTH,') of ',
                    'criterion variable:');
            GETVALIDINDEX;
            ERASE(18,3);
            GROUP[INDEX]:=1;
            P:=P+1;
            GROUP[-1]:=P;
            ROW:=INDEX+6;
            GOTOXY(39,ROW);
            WRITE(CHR(15),'CRITERION',CHR(14));
        END;  (* End of Assign Criterion *)

(*****************************************************************)

    PROCEDURE ASSIGNPREDICTOR;

        BEGIN
            WRITELN('Enter index (1-',WIDTH,') of ',
                    'predictor variable:');
            GETVALIDINDEX;
            ERASE(18,3);
            GROUP[INDEX]:=2;
            K:=K+1;
            GROUP[0]:=K;
            ROW:=INDEX+6;
            GOTOXY(39,ROW);
            WRITE(CHR(15),'PREDICTOR',CHR(14));
        END;  (* End of Assign Predictor *)

(*****************************************************************)

    PROCEDURE ASSIGNMANIFESTATION;

        BEGIN
            WRITELN('Enter index (1-',WIDTH,') of ',
                    'manifestation variable:');
            GETVALIDINDEX;
            ERASE(18,3);
            GROUP[INDEX]:=3;
            N:=N+1;
            GROUP[0]:=N;
            ROW:=INDEX+6;
            GOTOXY(39,ROW);
            WRITE(CHR(15),'SELECTED',CHR(14));
        END;  (* End of Assign Manifestation *)
```

```
(****************************************************************)

    PROCEDURE REMOVEASSIGNMENT;

        BEGIN
            WRITELN('Enter index (1-',WIDTH,') of ',
                    'variable to remove:');
            GETVALIDINDEX;

            IF (GROUP[INDEX]>0) AND (GROUP[INDEX]<4) THEN
                BEGIN
                    CASE (GROUP[INDEX]) OF
                        1: BEGIN
                               P:=P-1;
                               GROUP[-1]:=P;
                           END;
                        2: BEGIN
                               K:=K-1;
                               GROUP[0]:=K;
                           END;
                        3: BEGIN
                               N:=N-1;
                               GROUP[0]:=N;
                           END;
                    END;   (* End of Reduce CASE *)

                    ROW:=INDEX+6;
                    GOTOXY(39,ROW);
                    WRITE(' ':20);
                    GROUP[INDEX]:=0;
                END
            ELSE
                BEGIN
                    WRITE(CHR(7),'Sorry.  That variable isn''t ',
                         'selected.  Press any key to continue   ');
                    GETOPTION(OPT);
                    ERASE(18,5);
                END;
        END;   (* End of Remove Assignment *)

(****************************************************************)

    PROCEDURE ATTEMPTEXIT;

        BEGIN
            CASE (CAT) OF
                1: IF (P<2) OR (K<2) THEN
                       BEGIN
                           WRITELN(' ',CHR(15),'WARNING:',
                                   CHR(14),'  Must have at ',
                                   'least 2 variables of each ',
                                   'type.',CHR(13));
                           WRITELN(' ':11,'Press any key ',
                               'to continue');
                           GOTOXY(0,19);
                           GETOPTION(OPT);
                           ERASE(18,5);
                       END
                   ELSE
                       DONE:=TRUE;
```

```
          2: IF (N<2) THEN
                 BEGIN
                     WRITELN(' ',CHR(15),'WARNING:',
                             CHR(14),'  Must select at ',
                             'least 2 manifestation ',
                             'variables.',CHR(13));
                     WRITELN(' ':11,'Press any key ',
                         'to continue');
                     GOTOXY(0,19);
                     GETOPTION(OPT);
                     ERASE(18,5);
                 END
             ELSE
                 DONE:=TRUE;

      END;  (* End of CASE *)
  END;  (* End of ATTEMPT EXIT *)

(*******************************************************************)

    PROCEDURE GETCHOICE;

        BEGIN
            GOTOXY(0,18);
            WRITELN('Select desired option:');

            CASE (CAT) OF                    (* Display choices       *)
                1: BEGIN
                       WRITELN('        1 - Assign as Criterion');
                       WRITELN('        2 - Assign as Predictor');
                       WRITELN('        3 - Remove assignment');
                       WRITELN('        4 - Exit ASSIGN VARIABLES');
                   END;
                2: BEGIN
                       WRITELN('        1 - Assign as Manifestation');
                       WRITELN('        2 - Remove assignment');
                       WRITELN('        3 - Exit ASSIGN VARIABLES');
                   END;
            END;  (* End of Display Choice CASE *)

            GETOPTION(OPT);

            CASE (CAT) OF                    (* Accept choice         *)
                1: WHILE (OPT<'1') AND (OPT>'4') DO
                          GETOPTION(OPT);
                2: WHILE (OPT<'1') AND (OPT>'3') DO
                          GETOPTION(OPT);
            END;  (* End of Accept Choice CASE *)

            CASE (CAT) OF                    (* Change for correct    *)
                1: IF (OPT>'2') THEN
                       OPT:=CHR(ORD(OPT)+1);
                2: OPT:=CHR(ORD(OPT)+2);
            END;  (* End of Change for correct procedure call *)

            ERASE(18,5);
            GOTOXY(0,19);

        END;  (* End of GET assignment CHOICE *)
```

```
(*****************************************************************)
(*                  Main body of ASSIGN VARIABLES               *)
(*****************************************************************)

    BEGIN
        WIDTH:=SPECS2[0];                    (* Initialize parameters *)
        DONE:=FALSE;
        FEAS:=TRUE;
        P:=0;
        K:=0;
        N:=0;

        CHECKSIZE;                           (* Check size constraint *)

        IF (FEAS) THEN                       (* Continue if Feasible  *)
            BEGIN
                FOR I:=1 TO WIDTH DO
                    GROUP[I]:=0;

                GOTOXY(0,5);
                WRITELN(CHR(15),'FIELD NUMBER','WIDTH':9,'NAME':8,
                        ' ':10,'GROUP',CHR(14));

                DISPLAYSPECS;

                WHILE NOT(DONE) DO
                    BEGIN                    (* Make assignments      *)
                        GETCHOICE;

                        CASE (OPT) OF
                           '1':  ASSIGNCRITERION;
                           '2':  ASSIGNPREDICTOR;
                           '3':  ASSIGNMANIFESTATION;
                           '4':  REMOVEASSIGNMENT;
                           '5':  ATTEMPTEXIT;
                        END;  (* End of CASE *)

                    END;  (* End of Make Assignments *)
            END; (* End of Continue if Feasible *)
    END;  (* End of ASSIGN VARIABLES *)

(*****************************************************************)

PROCEDURE CALCULATE;

(*****************************************************************)
(*                                                               *)
(*       This procedure calculates and prints the MEAN and       *)
(*            STANDARD DEVIATION for designated variables in     *)
(*            GROUP (designated with other than zero).           *)
(*                                                               *)
(*****************************************************************)

    VAR
        I,                                  (* Iteration counter     *)
        INDEX:                              (* Field index into arrays *)
                INTEGER;
        OPT:                                (* Menu option           *)
                CHAR;
        NAME:                                   (* Field or variable name  *)
              STRING;

(*******************************************************************)
(*                        Internal Procedures                      *)
(*******************************************************************)

    PROCEDURE CALCXBAR;

        BEGIN
            FOR INDEX:=1 TO WIDTH DO        (* Calculate grand totals *)
                IF (GROUP[INDEX]>0) THEN
                    FOR I:=1 TO NUMREC DO
                        XBAR[INDEX]:=XBAR[INDEX]+DATA[I,INDEX];

            FOR INDEX:=1 TO WIDTH DO        (* Convert to means        *)
                IF (NUMREC=0) THEN
                    XBAR[INDEX]:=99.9999
                ELSE
                    XBAR[INDEX]:=XBAR[INDEX]/NUMREC;

        END;  (* End of CALCulate means (XBAR) *)

(*******************************************************************)

    PROCEDURE CALCSDEV;

        BEGIN
            FOR INDEX:=1 TO WIDTH DO       (* Calculate grand totals  *)
                IF (GROUP[INDEX]>0) THEN
                    IF (XBAR[INDEX]=99.9999) THEN
                        SDEV[INDEX]:=99.9999
                    ELSE
                        FOR I:=1 TO NUMREC DO
                            SDEV[INDEX]:=SDEV[INDEX]+
                                SQR(DATA[I,INDEX]-XBAR[INDEX]);

            FOR INDEX:=1 TO WIDTH DO       (* Convert to variances    *)
                IF (NUMREC<2) THEN
                    SDEV[INDEX]:=99.9999
                ELSE
                    SDEV[INDEX]:=SDEV[INDEX]/(NUMREC-1);

            FOR INDEX:=1 TO WIDTH DO       (* Convert to standard dev *)
                IF (NUMREC<2) THEN
                    SDEV[INDEX]:=99.9999
                ELSE
                    SDEV[INDEX]:=SQRT(SDEV[INDEX]);

        END;  (* End of CALCulate standard deviations (SDEV) *)

(*******************************************************************)

    PROCEDURE PRINTRESULTS;

        BEGIN
            GOTOXY(0,5);
            WRITELN(CHR(15),'VARIABLE':15,'MEAN':11,
                    'STANDARD DEVIATION':24,' ':6,CHR(14),CHR(13));
            IF (PRINTER) THEN
```

```
                BEGIN
                    WRITELN(PTR,'FILE ',SPECS1[0],':',CHR(13));
                    WRITELN(PTR,'VARIABLE':9,'MEAN':11,
                            'STANDARD DEVIATION':24,CHR(13));
                END;

            FOR I:=1 TO WIDTH DO        (* Print Criterion/Manifest. *)
                IF (GROUP[I]=1) OR (GROUP[I]=3) THEN
                    BEGIN
                        NAME:=SPECS1[I];
                        IF (LENGTH(NAME)>9) THEN
                            NAME:=COPY(NAME,1,9);
                        WRITELN(NAME:15,XBAR[I]:12:5,SDEV[I]:16:5);
                        IF (PRINTER) THEN
                            WRITELN(PTR,NAME:9,XBAR[I]:12:5,
                                    SDEV[I]:16:5);
                    END;  (* End of Print Criterion variables *)

            FOR I:=1 TO WIDTH DO        (* Print Predictor variables *)
                IF (GROUP[I]=2) THEN
                    BEGIN
                        NAME:=SPECS1[I];
                        IF (LENGTH(NAME)>9) THEN
                            NAME:=COPY(NAME,1,9);
                        WRITELN(NAME:15,XBAR[I]:12:5,SDEV[I]:16:5);
                        IF (PRINTER) THEN
                            WRITELN(PTR,NAME:9,XBAR[I]:12:5,
                                    SDEV[I]:16:5);
                    END;  (* End of Print Predictor variables *)

            IF (PRINTER) THEN
                FOR I:=1 TO 2 DO
                    WRITELN(PTR);

        END;  (* End of PRINT RESULTS *)

(*******************************************************************)
(*                        Main body of CALCULATE                   *)
(*******************************************************************)

    BEGIN
        FOR INDEX:=1 TO WIDTH DO        (* Zero out arrays          *)
            BEGIN
                XBAR[INDEX]:=0.0;
                SDEV[INDEX]:=0.0;
            END;

        CALCXBAR;                       (* Means of designated      *)

        CALCSDEV;                       (* Stand Dev of designated  *)

        ERASE(22,1);
        GOTOXY(16,22);
        WRITE(CHR(7),'Done.  Press any key to print results   ');
        GETOPTION(OPT);
        ERASE (22,1);

        PRINTRESULTS;                   (* XBAR's and SDEV's        *)

    END;  (* End of CALCULATE *)
```

```
(*******************************************************************)
(*                   Initialization part of UNIT                   *)
(*******************************************************************)

END.
```

```
(*$S+*)

UNIT MU_G; INTRINSIC CODE 17;

INTERFACE
    USES TRANSCEND, MAIN_UNIT;

    PROCEDURE STANDARDIZE(VAR DATA:RAWDATA;VAR XBAR,SDEV:VECTOR;
                          VAR GROUP:HEADER2;NUMREC,WIDTH:INTEGER;
                          OPTION:CHAR);

    PROCEDURE GENMATRIX(VAR DATA:RAWDATA;VAR CM:MATRIX;
                        VAR SPECS1:HEADER1;VAR GROUP:HEADER2;
                        NUMREC,WIDTH:INTEGER;PRINTER:BOOLEAN);

    PROCEDURE GETCVCS(VAR CC:VECTOR;VAR ALPHA,BETA,CM:MATRIX;
                      VAR SPECS1:HEADER1;VAR GROUP:HEADER2;
                      PRINTER:BOOLEAN);

IMPLEMENTATION

(******************************************************************)
(*                    Main body of MU_G                            *)
(******************************************************************)

PROCEDURE STANDARDIZE;

(******************************************************************)
(*                                                                 *)
(*       This procedure standardizes designated fields within      *)
(*            DATA depending on the value of OPTION:               *)
(*                                                                 *)
(*            1 - Mean Corrected (Subtract Mean only)              *)
(*            2 - Standardized    (Subtract Mean & divide by       *)
(*                                 Standard Deviation)             *)
(*                                                                 *)
(*       The first option leads to generation of a Sample          *)
(*           Covariance Matrix, the latter to a Sample             *)
(*           Correlation Matrix.                                   *)
(*                                                                 *)
(******************************************************************)

    VAR I,J:INTEGER;                       (* Iteration counters    *)

    BEGIN
        IF (OPTION='1') THEN               (* Subtract Means only   *)
            FOR J:=1 TO WIDTH DO           (*   of designated vars  *)
                IF (GROUP[J]>0) THEN
                    FOR I:=1 TO NUMREC DO
                        DATA[I,J]:=DATA[I,J]-XBAR[J];

        IF (OPTION='2') THEN               (* Sub Means & Div by SD *)
            FOR J:=1 TO WIDTH DO           (*   of designated vars  *)
                IF (GROUP[J]>0) THEN
                    FOR I:=1 TO NUMREC DO
                        IF (NUMREC<2) OR (SDEV[J]=0.0) THEN
                            DATA[I,J]:=99.9999
                        ELSE
                            DATA[I,J]:=(DATA[I,J]-XBAR[J])/SDEV[J];
    END;  (* End of STANDARDIZE *)

(***********************************************************************)

PROCEDURE GENMATRIX;

(***********************************************************************)
(*                                                                     *)
(*          This procedure generates and prints the Sample            *)
(*               Correlation  Matrix (CM) of the designated           *)
(*               fields by first generating smaller first and         *)
(*               second set self-correlation matrices and the         *)
(*               first/second cross-correlation matrix.               *)
(*               These partitions are stored in CORRMATRIX (CM).      *)
(*                                                                     *)
(***********************************************************************)

    VAR
        I,J,L,                                (* Iteration counters    *)
        INDEX,                                (* Field in arrays       *)
        P,                                    (* Number in 1st set     *)
        K:                                    (* Number in 2nd set     *)
               INTEGER;
        MULT:                                 (* Statistical reducer   *)
               REAL;
        OPT:                                  (* Menu option           *)
               CHAR;
        NAME:                                 (* Field / variable name *)
               STRING;
        A:                                    (* Array of pointers to  *)
               HEADER2;                       (*  next same type field *)

(***********************************************************************)
(*                         Internal Procedures                         *)
(***********************************************************************)

    PROCEDURE SETPOINTERS;

        BEGIN
            FOR I:=1 TO WIDTH DO
                IF (GROUP[I]=1) OR (GROUP[I]=3) THEN
                    BEGIN                           (* 1st or only set *)
                        INDEX:=INDEX+1;
                        A[INDEX]:=I;
                    END;

            FOR I:=1 TO WIDTH DO
                IF (GROUP[I]=2) THEN                (* 2nd set         *)
                    BEGIN
                        INDEX:=INDEX+1;
                        A[INDEX]:=I;
                    END;
        END;  (* End of SET sequential POINTERS by type *)

(***********************************************************************)

    PROCEDURE GETMATRIX;

        BEGIN
            FOR I:=1 TO P DO                      (* 1st set self-corr *)
            FOR J:=(I+1) TO P DO
                BEGIN
                    CM[I,J]:=0.0;
                    FOR L:=1 TO NUMREC DO
                        CM[I,J]:=CM[I,J]+
                                 (DATA[L,A[I]]*DATA[L,A[J]]);

                    CM[I,J]:=CM[I,J]*MULT;     (* Upper diagonal *)
                    CM[J,I]:=CM[I,J];          (* Lower diagonal *)
                END; (* End of 1st set self-correlation matrix *)

        FOR I:=(P+1) TO (P+K) DO           (* 2nd set self-corr *)
            FOR J:=(I+1) TO (P+K) DO
                BEGIN
                    CM[I,J]:=0.0;
                    FOR L:=1 TO NUMREC DO
                        CM[I,J]:=CM[I,J]+
                                 (DATA[L,A[I]]*DATA[L,A[J]]);

                    CM[I,J]:=CM[I,J]*MULT;     (* Upper diagonal *)
                    CM[J,I]:=CM[I,J];          (* Lower diagonal *)
                END; (* End of 2nd set self-correlation matrix *)

        FOR I:=1 TO P DO                   (* Cross correlation *)
            FOR J:=(P+1) TO (P+K) DO
                BEGIN
                    CM[I,J]:=0.0;
                    FOR L:=1 TO NUMREC DO
                        CM[I,J]:=CM[I,J]+
                                 (DATA[L,A[I]]*DATA[L,A[J]]);

                    CM[I,J]:=CM[I,J]*MULT;   (* First/Sec cross *)
                    CM[J,I]:=CM[I,J];        (* Sec/First cross *)
                END;  (* End of Cross correlation matrices     *)

        FOR I:=1 TO (P+K) DO             (* Main Diagonal to 1  *)
            CM[I,I]:=1.0;
    END;  (* End of GET the MATRIX *)

(*****************************************************************)

  PROCEDURE PRINTMATRIX;

    BEGIN
        GOTOXY(10,5);
        FOR I:=1 TO (P+K) DO             (* Display header row *)
            BEGIN
                NAME:=SPECS1[A[I]];
                IF (LENGTH(NAME)>7) THEN
                    NAME:=COPY(NAME,1,7);    (* Truncate to fit *)
                WRITE(NAME:8);
            END;
        WRITELN(CHR(13));

        IF (PRINTER) THEN                  (* Print header row  *)
            BEGIN
                WRITELN(PTR,'CORRELATION COEFFICIENTS:',CHR(13));
                WRITE(PTR,' ':10);
                FOR I:=1 TO (P+K) DO
                    BEGIN
                    NAME:=SPECS1[A[I]];
                    IF (LENGTH(NAME)>11) THEN
                         NAME:=COPY(NAME,1,11); (* Truncate *)
                    WRITE(PTR,NAME:12);
               END;
          WRITELN(PTR,CHR(13));
     END;  (* End of Display/Print Header Rows *)

FOR I:=1 TO (P+K) DO            (* Display/Print Matrix *)
     BEGIN
          NAME:=SPECS1[A[I]];
          IF (LENGTH(NAME)>9) THEN
               NAME:=COPY(NAME,1,9);          (* Truncate    *)
          WRITE(NAME:9,' ');
          IF (PRINTER) THEN
               WRITE(PTR,NAME:9,' ');

          FOR INDEX:=1 TO (P+K) DO
               BEGIN
                    WRITE(CM[I,INDEX]:8:4);
                    IF (PRINTER) THEN
                         WRITE(PTR,CM[I,INDEX]:12:4);
               END;

          WRITELN;
          IF (PRINTER) THEN
               WRITELN(PTR);
     END;  (* End of Display/Print Matrix *)

IF (PRINTER) THEN
     FOR I:=1 TO 2 DO
          WRITELN(PTR);

END;  (* End of PRINT the correlation MATRIX *)

(*****************************************************************)
(*                    Main body of GENMATRIX                     *)
(*****************************************************************)

BEGIN
     P:=GROUP[-1];                    (* Initialize parameters *)
     K:=GROUP[0];
     INDEX:=0;
     MULT:=1/(NUMREC-1);

     SETPOINTERS;                     (* Fill pointer array    *)

     GETMATRIX;

     ERASE(20,3);
     GOTOXY(16,22);
     WRITE(CHR(7),'Done.  Press any key to print results.    ');
     GETOPTION(OPT);
     ERASE(22,1);

     GOTOXY(0,2);
     WRITELN('CORRELATION COEFFICIENTS':28);

     PRINTMATRIX;
```

```
    END;  (* End of GENerate MATRIX *)

(*****************************************************************)

(*$I PSPP:GETCVCS *)                  (* Include procedure in memory *)

(*****************************************************************)
(*                  Initialization part of UNIT                  *)
(*****************************************************************)

END.
```

```
PROCEDURE GETCVCS;

(****************************************************************)
(*                                                              *)
(*      This procedure calculates the Canonical Variate         *)
(*           Coefficients (Alpha & Beta) for both sets of       *)
(*           variables (X & Y) and prints them.  Note: The      *)
(*           Alpha's are the normalized eigenvectors.           *)
(*                                                              *)
(*           ALPHA = 1/SQRT(C) * ALPHA                          *)
(*                   where C = ALPHA' * R(YY) * ALPHA           *)
(*                                                              *)
(*                                     -1                       *)
(*           BETA = (1/CANCOR) * (R(XX)   * R(XY)) * ALPHA      *)
(*                                                              *)
(****************************************************************)

    VAR
        I,J,L,                          (* Iteration counters      *)
        P,                              (* Number of criterions    *)
        K,                              (* Number of predictors    *)
        N:                              (* Lesser of P and K       *)
                INTEGER;
        SCALE:                          (* CVC Normalize factor    *)
                REAL;
        OPT:                            (* Menu option             *)
                CHAR;
        NAME:                           (* Variable name           *)
                STRING;
        NAMES:                          (* Sorted variable names   *)
                HEADER1;
        TEMP:                           (* Used in scaling         *)
                VECTOR;
        MMULT,                          (* Matrix multiplier       *)
        RXX,                            (* R(XX) partition         *)
        RXXINV:                         (* R(XX) inverse           *)
                MATRIX;
        FEAS:                           (* Used in INVERT routine  *)
                BOOLEAN;

(****************************************************************)
(*                     Internal Procedures                      *)
(****************************************************************)

    PROCEDURE INVERT(N:INTEGER;VAR FEAS:BOOLEAN;VAR R,IM:MATRIX);

        VAR
            I,J,                (* Iteration counters              *)
            ROW,                (* Pivot row                       *)
            COL:                (* Pivot column                    *)
                INTEGER;
            T,                  (* Pivot element on main diagonal  *)
            CM,                 (* Column multiplier               *)
            TR,                 (* Subtracted from matrix R        *)
            TIM:                (* Subtracted from matrix IM       *)
                REAL;

(****************************************************************)
(*                 Procedures internal to INVERT                *)
```

```
(*****************************************************************)

      PROCEDURE INITIALIZE;

         BEGIN
            FOR I:=1 TO N DO
               FOR J:=1 TO N DO
                  BEGIN
                     IM[I,J]:=0.0;
                     IF (I=J) THEN
                        IM[I,J]:=1.0;
                  END;
         END;   (* End of INITIALIZE *)

(*****************************************************************)

      PROCEDURE SCALEPIVOTROW;

         BEGIN
            FOR J:=1 TO N DO
               BEGIN
                  R[ROW,J]:=R[ROW,J]/T;
                  IM[ROW,J]:=IM[ROW,J]/T;
               END;
         END;   (* End of SCALE PIVOT ROW *)

(*****************************************************************)

      PROCEDURE REDUCEROWS;

         BEGIN
            CM:=-R[I,COL];
            FOR J:=1 TO N DO
               BEGIN
                  TR:=R[ROW,J]*CM;
                  TIM:=IM[ROW,J]*CM;

                  R[I,J]:=R[I,J]+TR;
                  IF (ABS(R[I,J])<0.000001) THEN
                     R[I,J]:=0.0;

                  IM[I,J]:=IM[I,J]+TIM;
                  IF (ABS(IM[I,J])<0.000001) THEN
                     IM[I,J]:=0.0;
               END;
         END;   (* End of REDUCE ROWS *)

(*****************************************************************)
(*                 Main body of INVERT Procedure                 *)
(*****************************************************************)

      BEGIN
         INITIALIZE;
         FEAS:=TRUE;

         FOR ROW:=1 TO N DO                  (* First scan         *)
            BEGIN
               COL:=ROW;
               T:=R[ROW,COL];
```

```
                IF (T<>0.0) THEN
                    BEGIN
                        IF (T<>1.0) THEN
                            SCALEPIVOTROW;

                        FOR I:=1 TO N DO
                            IF (I<>ROW) THEN
                                REDUCEROWS;
                    END;
            END;  (* End of First Scan *)

        FOR ROW:=1 TO N DO                  (* Second Scan        *)
            BEGIN
                COL:=ROW;
                T:=R[ROW,COL];

                IF (T=0.0) THEN
                    FEAS:=FALSE
                ELSE
                    IF (T<>1.0) THEN
                        SCALEPIVOTROW;

                FOR I:=1 TO N DO
                    IF (I<>ROW) AND (FEAS) THEN
                        REDUCEROWS;
            END;  (* End of Second Scan *)
    END;  (* End of INVERT *)

(****************************************************************)

  PROCEDURE SCALEALPHAS;

    BEGIN
        FOR L:=1 TO N DO                    (* Scale the Alpha's    *)
            BEGIN
                FOR I:=1 TO P DO
                    BEGIN
                        TEMP[I]:=0.0;
                        FOR J:=1 TO P DO
                                TEMP[I]:=TEMP[I]+
                                          ALPHA[J,L]*CM[J,I];
                    END;

                SCALE:=0.0;
                FOR I:=1 TO P DO
                    SCALE:=SCALE+TEMP[I]*ALPHA[I,L];

                FOR I:=1 TO P DO
                    IF (SCALE=0.0) OR (ALPHA[I,L]=99.9999) THEN
                        ALPHA[I,L]:=99.9999
                    ELSE
                        ALPHA[I,L]:=(1.0/SQRT(SCALE))*ALPHA[I,L];
            END;
    END;  (* End of SCALE the ALPHA vectorS *)

(****************************************************************)

  PROCEDURE CALCBETAS;

    BEGIN
```

```
            FOR I:=1 TO K DO                (* Get R(XX) inverse    *)
                FOR J:=1 TO K DO
                    RXX[I,J]:=CM[(I+P),(J+P)];

            INVERT(K,FEAS,RXX,RXXINV);
                                            (*          -1          *)
            FOR I:=1 TO K DO                (* Get R(XX)   * R(XY)  *)
                FOR J:=1 TO P DO
                    FOR L:=1 TO K DO
                        MMULT[I,J]:=MMULT[I,J]+
                                    (RXXINV[I,L]*CM[(L+P),J]);

            FOR I:=1 TO N DO                (* Calculate the BETAs *)
                FOR J:=1 TO K DO
                    IF (CC[I]=99.9999) OR (CC[I]=0.0) THEN
                        BETA[J,I]:=99.9999
                    ELSE
                        BEGIN
                            FOR L:=1 TO P DO
                                BETA[J,I]:=BETA[J,I]+
                                           (MMULT[J,L]*ALPHA[L,I]);
                            BETA[J,I]:=(1.0/CC[I])*BETA[J,I],
                        END;
        END;  (* End of CALCulate the BETA vectorS *)

(******************************************************************)

    PROCEDURE PRINTFIRSTSET;

        BEGIN
            GOTOXY(0,3);
            WRITELN('COEFFICIENTS FOR CANONICAL VARIABLES OF ',
                    'THE FIRST SET');
            GOTOXY(0,5);
            WRITE(' ':15);
            FOR I:=1 TO N DO
                WRITE('  CANVAR',I:2);

            IF (PRINTER) THEN               (* Printer headings     *)
                BEGIN
                    WRITELN(PTR,'COEFFICIENTS FOR CANONICAL ',
                            'VARIABLES OF THE FIRST SET',CHR(13));
                    WRITE(PTR,' ':15);
                    FOR I:=1 TO N DO
                        WRITE(PTR,'  CANVAR',I:2);
                    WRITELN(PTR,CHR(13));
                END;

            GOTOXY(0,7);                 (* Get NAMEs of first set *)
            J:=0;
            FOR I:=1 TO (P+K) DO
                IF (GROUP[I]=1) THEN
                    BEGIN
                        J:=J+1;
                        NAMES[J]:=SPECS1[I];
                    END;

            FOR I:=1 TO P DO                (* Print first set      *)
                BEGIN
                    NAME:=NAMES[I];
```

```
                  IF (LENGTH(NAME)>15) THEN
                      NAME:=COPY(NAME,1,15);
                  WRITE(NAME:15);
                  IF (PRINTER) THEN
                      WRITE(PTR,NAME:15);

                  FOR J:=1 TO N DO
                      BEGIN
                          WRITE(ALPHA[I,J]:10:4);
                          IF (PRINTER) THEN
                              WRITE(PTR,ALPHA[I,J]:10:4);
                      END;

                  WRITELN;
                  IF (PRINTER) THEN
                      WRITELN(PTR);
              END;
      END;  (* End of PRINT the FIRST SET of coefficients *)

(*****************************************************************)

    PROCEDURE PRINTSECONDSET;

      BEGIN
          WRITELN(CHR(13),'COEFFICIENTS FOR CANONICAL VARIABLES ',
                 'OF THE SECOND SET',CHR(13));
          WRITE(' ':15);
          FOR I:=1 TO N DO
              WRITE('  CANVAR',I:2);
          WRITELN(CHR(13));

          IF (PRINTER) THEN               (* Printer headings     *)
              BEGIN
                  WRITELN(PTR,CHR(13),'COEFFICIENTS FOR CANONICAL',
                         ' VARIABLES OF THE SECOND SET',CHR(13));
                  WRITE(PTR,' ':15);
                  FOR I:=1 TO N DO
                      WRITE(PTR,'  CANVAR',I:2);
                  WRITELN(PTR,CHR(13));
              END;

          J:=0;                        (* Get NAMES of second set *)
          FOR I:=1 TO (P+K) DO
              IF (GROUP[I]=2) THEN
                  BEGIN
                      J:=J+1;
                      NAMES[J]:=SPECS[I];
                  END;

          FOR I:=1 TO K DO                (* Print second set     *)
              BEGIN
                  NAME:=NAMES[I];
                  IF (LENGTH(NAME)>15) THEN
                      NAME.=COPY(NAME,1,15);
                  WRITE(NAME:15);
                  IF (PRINTER) THEN
                      WRITE(PTR,NAME:15);

                  FOR J:=1 TO N DO
                      BEGIN
```

```
                              WRITE(BETA[I,J]:10:4);
                              IF (PRINTER) THEN
                                  WRITE(PTR,BETA[I,J]:10:4);
                          END;

                      WRITELN;
                      IF (PRINTER) THEN
                          WRITELN(PTR);
                  END;

              IF (PRINTER) THEN
                  WRITELN(PTR,CHR(13));
          END;   (* End of PRINT the SECOND SET of coefficients *)

(*******************************************************************)
(*                         Main body of GETCVCS                    *)
(*******************************************************************)

    BEGIN
    (*$R TRANSCEND *)                        (* Retain UNIT in memory    *)
        P:=GROUP[-1];                        (* Initialize parameters    *)
        K:=GROUP[0];
        IF (P>K) THEN
            N:=K
        ELSE
            N:=P;

        FOR I:=1 TO K DO
            FOR J:=1 TO P DO
                BEGIN
                    MMULT[I,J]:=0.0;
                    BETA[I,J]:=0.0;
                END;

        SCALEALPHAS;                         (* Scale ALPHA vectors      *)

        CALCBETAS;                           (* Calculate BETA vectors   *)

        ERASE(22,1);
        GOTOXY(16,22);
        WRITE('Done.  Press any key to print results.   ');
        GETOPTION(OPT);
        ERASE(22,1);

        PRINTFIRSTSET;

        PRINTSECONDSET;

        GOTOXY(16,22);
        WRITE('Done.  Press any key to continue.   ');
        GETOPTION(OPT);
        ERASE(3,20);
    END;  (* End of GET Canonical Variate CoefficientS *)

(*******************************************************************)
```

```
(*$S+*)

UNIT MU_H; INTRINSIC CODE 18;

INTERFACE
    USES MAIN_UNIT;

    PROCEDURE INVERT(N:INTEGER;VAR FEAS:BOOLEAN;
                     R:MATRIX;VAR IM:MATRIX);

    PROCEDURE PREFTOEIG(CM:MATRIX;P,K:INTEGER;VAR A:MATRIX;
                        VAR FEAS:BOOLEAN);

    PROCEDURE EIGEN(N:INTEGER;A:MATRIX;VAR V:MATRIX;VAR E:VECTOR);

IMPLEMENTATION

(*****************************************************************)
(*                      Main body of MU_H                        *)
(*****************************************************************)

PROCEDURE INVERT;

(*****************************************************************)
(*                                                               *)
(*      This procedure needs as input:                           *)
(*                                                               *)
(*           N - Order of matrix to be inverted                  *)
(*           R(N,N) - Matrix to be inverted                      *)
(*                                                               *)
(*      This procedure returns as output:                        *)
(*                                                               *)
(*           IM(N,N) - Inverted matrix                           *)
(*           FEAS - Returns FALSE if no inverse exists           *)
(*                                                               *)
(*****************************************************************)

    VAR
        I,J,                       (* Iteration counters            *)
        ROW,                       (* Pivot row                     *)
        COL:                       (* Pivot column                  *)
           INTEGER;
        T,                         (* Pivot element on main diagonal *)
        CM,                        (* Column multiplier             *)
        TR,                        (* Subtracted from matrix R      *)
        TIM:                       (* Subtracted from matrix IM     *)
           REAL;

(*****************************************************************)
(*                     Internal Procedures                       *)
(*****************************************************************)

    PROCEDURE INITIALIZE;

        BEGIN
           FOR I:=1 TO N DO
              FOR J:=1 TO N DO
                 BEGIN
                    IM[I,J]:=0.0;
                    IF (I=J) THEN
                    IM[I,J]:=1.0;
                END;
        END;  (* End of INITIALIZE *)

(****************************************************************)

    PROCEDURE SCALEPIVOTROW;

        BEGIN
            FOR J:=1 TO N DO
                BEGIN
                    R[ROW,J]:=R[ROW,J]/T;
                    IM[ROW,J]:=IM[ROW,J]/T;
                END;
        END;  (* End of SCALE PIVOT ROW *)

(****************************************************************)

    PROCEDURE REDUCEROWS;

        BEGIN
            CM:=-R[I,COL]:
            FOR J:=1 TO N DO
                BEGIN
                    TR:=R[ROW,J]*CM;
                    TIM:=IM[ROW,J]*CM;

                    R[I,J]:=R[I,J]+TR;
                    IF (ABS(R[I,J])<0.000001) THEN
                        R[I,J]:=0.0;

                    IM[I,J]:=IM[I,J]+TIM;
                    IF (ABS(IM[I,J])<0.000001) THEN
                        IM[I,J]:=0.0;
                END;
        END;  (* End of REDUCE ROWS *)

(****************************************************************)
(*                  Main body of INVERT Procedure               *)
(****************************************************************)

    BEGIN
        INITIALIZE;
        FEAS:=TRUE;

        FOR ROW:=1 TO N DO                         (* First scan      *)
            BEGIN
                COL:=ROW;
                T:=R[ROW,COL];

                IF (T<>0.0) THEN
                    BEGIN
                        IF (T<>1.0) THEN
                            SCALEPIVOTROW;

                        FOR I:=1 TO N DO
                            IF (I<>ROW) THEN
                                REDUCEROWS;
                    END;
            END;  (* End of First Scan *)
        FOR ROW:=1 TO N DO                              (* Second Scan       *)
            BEGIN
                COL:=ROW;
                T:=R[ROW,COL];

                IF (T=0.0) THEN
                    FEAS:=FALSE
                ELSE
                    IF (T<>1.0) THEN
                        SCALEPIVOTROW;

                FOR I:=1 TO N DO
                    IF (I<>ROW) AND (FEAS) THEN
                        REDUCEROWS;
            END;  (* End of Second Scan *)
    END;  (* End of INVERT *)

(*********************************************************************)

PROCEDURE PREPTOEIG;

(*********************************************************************)
(*                                                                   *)
(*        This procedure generates matrix A, from which the          *)
(*             eigenvalues are calculated, by multiplying the        *)
(*             partitions of the Sample Correlation Matrix:          *)
(*                                                                   *)
(*                        -1                -1                       *)
(*              A = [ R(YY)   * R(YX) * R(XX)   * R(XY) ]            *)
(*                                                                   *)
(*********************************************************************)

    VAR  I,J,L:  INTEGER;                       (* Iteration counters  *)

(*********************************************************************)
(*                         Internal Procedure                        *)
(*********************************************************************)

    PROCEDURE CLEAR(VAR A:MATRIX);              (* Empty matrix A      *)

        BEGIN
            FOR I:=1 TO MAXSIZE DO
                FOR J:=1 TO MAXSIZE DO
                    A[I,J]:=0.0;
        END;

(*********************************************************************)

    PROCEDURE GETUPPERLEFT;

        BEGIN
            FOR I:=1 TO P DO                    (* Get R(YY)' * R(YX)  *)
                FOR J:=1 TO K DO
                    FOR L:=1 TO P DO
                        A[I,J]:=A[I,J]+(CM[I,L]*CM[L,(J+P)]);

            FOR I:=1 TO P DO
                FOR J:=1 TO K DO
                    CM[I,(J+P)]:=A[I,J];
```

```
          CLEAR(A);
     END;

(****************************************************************)

   PROCEDURE GETLOWERRIGHT;

     BEGIN
          FOR I:=1 TO K DO                  (* Get R(XX)' * R(XY)   *)
               FOR J:=1 TO P DO
                    FOR L:=1 TO K DO
                         A[I,J]:=A[I,J]+
                              (CM[(I+P),(L+P)]*CM[(L+P),J]);

          FOR I:=1 TO K DO
               FOR J:=1 TO P DO
                    CM[(I+P),J]:=A[I,J];
          CLEAR(A);
     END;

(****************************************************************)

   PROCEDURE GETMATRIXA;

     BEGIN
          FOR I:=1 TO P DO
               FOR J:=1 TO P DO
                    FOR L:=1 TO K DO
                         A[I,J]:=A[I,J]+(CM[I,(L+P)]*CM[(L+P),J]);
     END;

(****************************************************************)
(*                    Main body of PREPTOEIG                    *)
(****************************************************************)

   BEGIN
     FEAS:=TRUE;

     FOR I:=1 TO P DO                       (* Get R(YY)'           *)
          FOR J:=1 TO P DO
               A[I,J]:=CM[I,J];

     INVERT(P,FEAS,A,A);

     IF (FEAS) THEN                         (* Proceed if feasable *)
          BEGIN
               FOR I:=1 TO P DO
                    FOR J:=1 TO P DO
                         CM[I,J]:=A[I,J];

               CLEAR(A);

               FOR I:=1 TO K DO             (* Get R(XX)'           *)
                    FOR J:=1 TO K DO
                         A[I,J]:=CM[(I+P),(J+P)];

               INVERT(K,FEAS,A,A);

               IF (FEAS) THEN               (* Proceed if feasable *)
                    BEGIN
```

```
                        FOR I:=1 TO K DO
                            FOR J:=1 TO K DO
                                CM[(I+P),(J+P)]:=A[I,J];

                        CLEAR(A);

                        GETUPPERLEFT;

                        GETLOWERRIGHT;

                        GETMATRIXA;

                    END;  (* End of 2nd FEASability check *)
            END;  (* End of 1st FEASability check *)
    END;  (* End of PREPare TO EIGen *)

(*********************************************************************)

PROCEDURE EIGEN;

(*********************************************************************)
(*                                                                   *)
(*        This procedure needs as input:                             *)
(*                                                                   *)
(*             N - Order of matrix to be solved                      *)
(*             A(N,N) - Matrix to find eigenvalues of                *)
(*                                                                   *)
(*        This procedure returns as output:                          *)
(*                                                                   *)
(*             E(N) - Eigenvalues in decreasing order                *)
(*             V(N,N) - Associated eigenvectors                      *)
(*                                                                   *)
(*********************************************************************)

    VAR
        UR,                  (* Top row of deflated matrices         *)
        Z:                   (* Eigenvectors of deflated matrices    *)
              MATRIX;
        B,                   (* Current eigenvector                  *)
        C:                   (* Dot product of A(N,N) and B(N)       *)
              VECTOR;
        I,J,                 (* Iteration counters                   *)
        L,                   (* Eigenvalue counter                   *)
        NS:                  (* Top row/Left col of deflated matrix  *)
              INTEGER;
        VALUE,               (* Current estimation of eigenvalue     *)
        LAST,                (* Previous estimation of eigenvalue    *)
        EPS,                 (* Accuracy stopping criteria           *)
        SCALE:               (* Eigenvector scale factor             *)
              REAL;

(*********************************************************************)
(*                        Internal Procedures                        *)
(*********************************************************************)

    PROCEDURE INITIALIZE;

        BEGIN                            (* Initialize matrices      *)
            FOR I:=1 TO N DO
                BEGIN
```

```
                    FOR J:=1 TO N DO
                        BEGIN
                            V[I,J]:=0.0;
                            UR[I,J]:=0.0;
                            Z[I,J]:=0.0;
                        END;
                    B[I]:=0.0;
                    UR[1,I]:=A[1,I];
                END;
            B[1]:=1.0;
        END;  (* End of INITIALIZE *)

(*******************************************************************)

    PROCEDURE GETEPS;

        VAR OPT1,OPT2,OPT3: CHAR;           (* Menu options         *)

        BEGIN                               (* Get stopping value   *)
            OPT2:='Y';
        (*$I-*)                             (* Turn off I/O check   *)
            GOTOXY(0,6);
            WRITELN('Do you desire optional precision setting?');
            WRITELN('    (More precision requires more time)');

            WHILE (OPT2='Y') OR (OPT2='y') DO
                BEGIN
                    GOTOXY(0,10);
                    WRITELN('Select desired option:');
                    WRITELN('        1 - Enter desired EPSILON ',
                            'value');
                    WRITELN('        2 - Go with default of 0.0001');
                    GETOPTION(OPT1);
                    WHILE (OPT1<>'1') AND (OPT1<>'2') DO
                        GETOPTION(OPT1);

                    IF (OPT1='1') THEN
                        BEGIN
                            ERASE(10,3);
                            GOTOXY(0,10);
                            WRITELN('Enter desired EPSILON value:');
                            WRITELN(CHR(7));
                            RESET(INPUT);
                            READ(EPS);

                            WHILE (IORESULT=14) OR (EPS>0.1) OR
                                  (EPS<0.000001) DO
                                BEGIN
                                    GOTOXY(1,20);
                                    WRITE(CHR(15),'WARNING:',
                                            CHR(14),'  Must be ',
                                           'between .000001 ',
                                           'and .1.  Press any ',
                                           'key to try again.');
                                    GOTOXY(0,20);
                                    GETOPTION(OPT3);
                                    ERASE(12,11);
                                    GOTOXY(0,12);
                                    RESET(INPUT);
                                    READ(EPS);
```

```
                              END;
                          END
                      ELSE
                          EPS:=0.0001;

                      GOTOXY(0,10);
                      WRITELN('EPSILON setting is',EPS:9:6,
                             ' between iterations.');
                      WRITELN(' ':40);
                      WRITELN('Do you want to make a change?',' ':10);
                      WRITELN('    Y - Yes, go back and change it');
                      WRITELN('    N - No, stay with this value');
                      GETOPTION(OPT2);
                      WHILE (OPT2<>'Y') AND (OPT2<>'N') AND
                            (OPT2<>'y') AND (OPT2<>'n') DO
                          GETOPTION(OPT2);
                      ERASE(10,5);
                  END;  (* End of WHILE loop *)
              (*$I+*)                          (* Turn on I/O checking *)
              ERASE(6,2);
          END;  (* End of GET EPSilon *)

(******************************************************************)

      PROCEDURE PREPARE;

          FUNCTION MAX(A,B:REAL):REAL;  (* Get max offset from zero  *)
              BEGIN
                  IF (ABS(A)>ABS(B)) THEN
                      MAX:=A
                  ELSE
                      MAX:=B;
              END;  (* End of MAX *)

          BEGIN                            (* Prepare to get eigenvalue *)
              SCALE:=0.0;

              FOR I:=NS TO N DO
                  BEGIN
                      C[I]:=0.0;
                      FOR J:=NS TO N DO
                          C[I]:=C[I]+A[I,J]*B[J];
                      SCALE:=MAX(SCALE,C[I]);
                  END;

              FOR I:=NS TO N DO
                  IF (SCALE=0.0) THEN
                      C[I]:=0.0
                  ELSE
                      C[I]:=C[I]/SCALE;
          END;  (* End of PREPARE to get eigenvalue *)

(******************************************************************)

      PROCEDURE GETEIGEN;

          VAR X,Y:  REAL;

          BEGIN                             (* Estimate eigenvalue      *)
              X:=0.0;
```

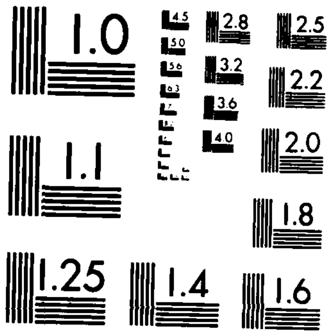
```
            Y:=0.0;

            FOR I:=NS TO N DO
                BEGIN
                    X:=X+B[I]*B[I];
                    Y:=Y+B[I]*C[I];
                END;

            IF (X=0.0) THEN
                VALUE:=99.9999
            ELSE
                VALUE:=SCALE*(Y/X);

            IF (VALUE<0.0) THEN
                VALUE:=0.0;
        END;  (* End of GET EIGENvalue *)

(*******************************************************************)

    PROCEDURE GETNEWVECTOR;

        VAR SUM,T:  REAL;

        BEGIN                          (* Eigenvectors of original   *)
            E[L]:=VALUE;

            FOR I:=NS TO N DO
                IF (B[NS]=0.0) THEN
                    C[I]:=0.0
                ELSE
                    C[I]:=B[I]/B[NS];

            FOR I:=(L-1) DOWNTO 1 DO
                BEGIN
                    SUM:=0.0;

                    FOR J:=1 TO N DO
                        SUM:=SUM+UR[I,J]*B[J];

                    IF (SUM<>0.0) THEN
                        BEGIN
                            T:=(VALUE-E[I])/SUM;
                            FOR J:=1 TO N DO
                                B[J]:=Z[I,J]+T*B[J];
                        END;
                END;

            FOR I:=1 TO N DO
                IF (E[L]=0.0) OR (E[L]=99.9999) THEN
                    V[I,L]:=99.9999
                ELSE
                    V[I,L]:=B[I];
        END;  (* End of GET NEW VECTOR *)

(*******************************************************************)

    PROCEDURE REDUCEMATRIX;

        VAR DF:  MATRIX;
        BEGIN                             (* Deflate original matrix *)
            FOR I:=NS TO N DO
                FOR J:=NS TO N DO
                    DF[I,J]:=C[I]*A[NS,J];

            FOR I:=NS TO N DO
                FOR J:=NS TO N DO
                    A[I,J]:=A[I,J]-DF[I,J];

            NS:=NS+1;

            FOR I:=NS TO N DO
                UR[NS,I]:=A[NS,I];

            FOR I:=1 TO (NS-1) DO
                B[I]:=0.0;

            B[NS]:=1.0;
        END;  (* End of REDUCE MATRIX *)

(*******************************************************************)

    FUNCTION DONE(A,B,EPS:REAL):BOOLEAN;

        BEGIN
            DONE:=(ABS(A-B) <= EPS);
        END;

(*******************************************************************)
(*                   Main body of EIGEN procedure                   *)
(*******************************************************************)

    BEGIN
        INITIALIZE;
        NS:=1;
        GETEPS;

        FOR L:=1 TO N DO                    (* Get each eigenvalue    *)
            BEGIN
                LAST:=0.0;
                VALUE:=N;

                WHILE NOT(DONE(LAST,VALUE,EPS)) DO
                    BEGIN
                        LAST:=VALUE;

                        PREPARE;

                        GETEIGEN;

                        FOR I:=NS TO N DO          (* Scale vector *)
                            IF (C[NS]=0.0) THEN
                                B[I]:=0.0
                            ELSE
                                B[I]:=C[I]/C[NS];
                    END;  (* End of WHILE loop *)

                FOR I:=1 TO N DO            (* Save each eigenvector *)
                    Z[L,I]:=B[I];
                    GETNEWVECTOR;

                    IF (L<N) THEN
                        REDUCEMATRIX;

               END;  (* End of Each Eigenvalue *)
     END;  (* End of get EIGENvalue *)

(*******************************************************************)
(*                    Initialization part of UNIT                  *)
(*******************************************************************)

END.
```

```
(*$S+*)

UNIT MU_I; INTRINSIC CODE 19;

INTERFACE
    USES TRANSCEND, APPLESTUFF, MAIN_UNIT;

    PROCEDURE USERSELECT(VAR N,NS:INTEGER);

    PROCEDURE BARTLETT(VAR N,NS,NUMREC:INTEGER;VAR EIGVAL:VECTOR);

    PROCEDURE SCREE(VAR N,NS:INTEGER;VAR EIGVAL:VECTOR);

    PROCEDURE SELECTFACTORS(VAR NS,NUMREC:INTEGER:VAR EIGVAL:VECTOR;
                            VAR GROUP:HEADER2;PRINTER:BOOLEAN);

IMPLEMENTATION

(*******************************************************************)
(*                         Main part of MU_I                       *)
(*******************************************************************)

PROCEDURE USERSELECT;

(*******************************************************************)
(*                                                                 *)
(*        This procedure asks the user the number of available     *)
(*             factors that should be kept for analysis.           *)
(*                                                                 *)
(*******************************************************************)

    VAR OPT: CHAR;                          (* Menu option          *)

    BEGIN
    (*$I-*)
        GOTOXY(0,18);
        WRITELN(CHR(7),'Enter number of factors (1-',N,') to ',
                'keep:',CHR(13));
        RESET(INPUT);
        READ(NS);

        WHILE (IORESULT=14) OR (NS<1) OR (NS>N) DO
            BEGIN
                GOTOXY(1,21);
                WRITELN(CHR(15),'WARNING:',CHR(14),
                        '  Must keep at least 1 and no more ',
                        'than ',N,' factors.');
                WRITELN(' ':11,'Press any key to try again');
                GOTOXY(0,21);
                GETOPTION(OPT);
                ERASE(20,3);
                GOTOXY(0,20);
                RESET(INPUT);
                READ(NS);
            END;  (* End of Bad number of factors *)
        ERASE(18,3);
    (*$I+*)
    END;  (* End of USER SELECT *)

(*******************************************************************)
```

```
PROCEDURE BARTLETT;

(*****************************************************************)
(*                                                               *)
(*       This procedure calculates the CHI-Square statistic      *)
(*            for the Bartlett test of significance for as       *)
(*            many of the factors as the user desires.  The      *)
(*            user is then asked to select the number of         *)
(*            significant factors to keep for FACTOR analysis.   *)
(*                                                               *)
(*****************************************************************)

    VAR
        I,                                    (* Iteration counter       *)
        INDEX,                                (* Non-zero factors        *)
        ROW:                                  (* Row on screen           *)
                INTEGER;
        R,W,                                  (* Parts of the statistic  *)
        MULT1,                                (* Statistical multipliers *)
        MULT2,
        STAT:                                 (* Bartlett statistic      *)
                REAL;
        OPT:                                  (* Menu option             *)
                CHAR;
        DONE:                                 (* Completion indicator    *)
                BOOLEAN;

(*****************************************************************)
(*                    Internal procedures                        *)
(*****************************************************************)

    PROCEDURE INSTRUCTUSER;

        BEGIN
            WRITE(CHR(12));
            GOTOXY(26,1);
            WRITELN(CHR(15),' BARTLETT SPHERICITY TEST ',CHR(14));

            GOTOXY(0,4);
            WRITELN('This test calculates a CHI-Square test ',
                    'statistic to check the hypothesis:',CHR(13));
            WRITELN('      Ho: EIGVAL(r+1)=EIGVAL(r+2)=...=',
                    'EIGVAL(K)=0');
            WRITELN('         vs.');
            WRITELN('      Ha: EIGVAL(r+1) <> 0;  after ''r'' tests');
            WRITELN;
            WRITELN('You should reference a CHI-Square table for');
            WRITELN;
            WRITELN('      CHI [ a , (k-r)(k-r-1) ], at level a');
            WRITELN;
            WRITELN('This routine calculates one test statistic, ',
                    'then asks if you wish to continue.');
            WRITELN('You must decide when to stop.  It will ',
                    'stop automatically after calculating');
            WRITELN(N,' values.  You will then be asked to select ',
                  'the number of factors to keep.',CHR(13));
            WRITELN('NOTE:  This test is good for small samples ',
                    '(n < 100) or for a large');
            WRITELN('        number of manifestation variables ',
```

AD-A141 049 PASCAL STATISTICAL PROCEDURES PACKAGE (PSPP)(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGINEERING D P KUNKEL DEC 83 AFIT/GSO/OS/83D-4 3/3

UNCLASSIFIED F/G 9/2 NL


1.0
2.8
2.5
3.2
2.2
3.6
4.0
2.0
1.1
1.8
1.25
1.4
1.6

```
                '(k > 9).');
          GOTOXY(22,22);
          WRITE('Press any key to start routine    ');
          GETOPTION(OPT);
          ERASE(4,19);

     END;  (* End of INSTRUCT USER *)

(*****************************************************************)

    FUNCTION DIVISOR(R:INTEGER):REAL;

        VAR TEMP1,TEMP2:REAL;              (* Temporary variables    *)

        BEGIN
            TEMP1:=1.0;
            TEMP2:=N/(N-R);
            FOR I:=1 TO R DO
                TEMP2:=TEMP2-(EIGVAL[I])/(N-R);
            FOR I:=1 TO (N-R) DO
                TEMP1:=TEMP1*TEMP2;
            DIVISOR:=TEMP1;
        END;

(*****************************************************************)
(*                    Main body of BARTLETT                      *)
(*****************************************************************)

    BEGIN
    (*$R TRANSCEND *)                  (* Retain UNIT in memory      *)
        R:=1.0;                        (* Initialize parameters      *)
        DONE:=FALSE;
        MULT1:=-(NUMREC - 1 - (2*N+5)/6.0);
        MULT2:=-((NUMREC+1) - (2*N+5)/6.0);

        INSTRUCTUSER;

        FOR INDEX:=1 TO N DO
            R:=R*EIGVAL[INDEX];

        W:=R;                          (* Calculate 1st statistic    *)
        IF (W=0.0) THEN
            STAT:=99.9999
        ELSE
            STAT:=MULT1*LN(W);

        GOTOXY(0,8);
        WRITELN('Overall Statistic = ',STAT:7:4);
        INDEX:=0;

        WHILE NOT(DONE) DO             (* Get sequential statistics *)
            BEGIN
                GOTOXY(0,20);
                WRITELN('Select desired option:');
                WRITELN('        1 - Continue with sequential ',
                        'test(s)');
                WRITELN('        2 - Exit & Choose number of ',
                        'factors');
                GETOPTION(OPT);
                WHILE (OPT<>'1') AND (OPT<>'2') DO
                GETOPTION(OPT);
            ERASE(20,3);

            ROW:=INDEX+10;
            GOTOXY(0,ROW);

            IF (OPT='2') THEN
                DONE:=TRUE;

            IF NOT(DONE) THEN
                BEGIN
                    INDEX:=INDEX+1;
                    R:=R/EIGVAL[INDEX];
                    W:=R/DIVISOR(INDEX);
                    STAT:=MULT2*LN(W);
                    WRITELN('  With ',INDEX,' non-zero = ',
                            STAT:7:4);
                END;

            IF (INDEX=N-1) THEN
                DONE:=TRUE;
        END;  (* End of Sequential test *)

    USERSELECT(N,NS);                  (* Get number of factors  *)

  END;  (* End of BARTLETT *)

(*****************************************************************)

PROCEDURE SCREE;

(*****************************************************************)
(*                                                               *)
(*       This procedure displays instructions on how to select   *)
(*            the number of factors to keep based on a plot of   *)
(*            Eigenvalue Magnitude and Factor Number.  The       *)
(*            user is then shown the plot and asked to select    *)
(*            the number (NS) of significant factors to keep.    *)
(*                                                               *)
(*****************************************************************)

    VAR
      INDEX,                        (* Index into EIGVAL array    *)
      I,                            (* Iteration counter          *)
      PITCH,                        (* Sound used in curve plot   *)
      ROW,                          (* Row for data display       *)
      COL,                          (* Column for data display    *)
      X,Y:                          (* Positions on text screen   *)
           INTEGER;
      VALUE:                        (* Eigenvalue being plotted   *)
           REAL;
      OPT:                          (* Menu option                *)
           CHAR;
      LETTER:                       (* Used in printing the labels *)
           STRING;

(*****************************************************************)
(*                    Internal Procedures                        *)
(*****************************************************************)
```

```
PROCEDURE DRAWAXIS;

    BEGIN
        FOR Y:=0 TO 22 DO           (* Vertical Axis                *)
            BEGIN
                GOTOXY(10,Y);
                WRITE(CHR(15),'  ',CHR(14));
            END;

        FOR X:=8 TO 52 DO           (* Horizontal Axis              *)
            BEGIN
                GOTOXY(X,21);
                WRITE(CHR(15),' ',CHR(14));
            END;

        FOR X:=50 DOWNTO 14 DO      (* Horizontal hash marks        *)
            IF ((X-14) MOD 4 = 0) THEN
                BEGIN
                    GOTOXY(X,21);
                    WRITE('  ');
                END;

        FOR Y:=21 DOWNTO 1 DO       (* Vertical hash marks          *)
            IF ((Y-1) MOD 2 = 0) THEN
                BEGIN
                    GOTOXY(10,Y);
                    WRITE('  ');
                END;
    END;  (* End of DRAW AXIS *)

(*********************************************************************)

PROCEDURE LABELAXIS;

    BEGIN
        IF (N>5) THEN             (* Vertical scale: Compressed   *)
            BEGIN
                ROW:=10;
                FOR Y:=1 TO 19 DO
                    IF ((Y-1) MOD 2 = 0) THEN
                        BEGIN
                            GOTOXY(7,Y);
                            WRITE(ROW:2);
                            ROW:=ROW-1;
                        END;
            END
        ELSE
            BEGIN                 (* Vertical scale: Expanded     *)
                ROW:=5;
                FOR Y:=1 TO 19 DO
                    IF ((Y-1) MOD 4 = 0) THEN
                        BEGIN
                            GOTOXY(7,Y);
                            WRITE(ROW:2);
                            ROW:=ROW-1;
                        END;
            END;  (* End of Vertical Scale *)

        IF (N>5) THEN             (* Horizontal scale: Compressed *)
            BEGIN
```

```
                    COL:=1;
                    FOR X:=14 TO 50 DO
                         IF ((X-14) MOD 4 = 0) THEN
                              BEGIN
                                   GOTOXY(X,22);
                                   WRITE(COL:2);
                                   COL:=COL+1;
                              END;
               END
          ELSE
               BEGIN                         (* Horizontal scale: Expanded    *)
                    COL:=1;
                    FOR X:=14 TO 50 DO
                         IF ((X-18) MOD 8 = 0) THEN
                              BEGIN
                                   GOTOXY(X,22);
                                   WRITE(COL:2);
                                   COL:=COL+1;
                              END;
               END;  (* End of Horizontal scale *)

          Y:=0;
          FOR I:=1 TO 11 DO             (* Vertical label                   *)
               BEGIN
                    LETTER:=COPY('EIGENVALUES',I,1);
                    GOTOXY(3,Y);
                    WRITE(LETTER);
                    Y:=Y+2;
               END;

          X:=21;
          FOR I:=1 TO 7 DO              (* Horizontal label                 *)
               BEGIN
                    LETTER:=COPY('FACTORS',I,1);
                    GOTOXY(X,23);
                    WRITE(LETTER);
                    X:=X+4;
               END;
     END;  (* End of LABEL AXIS *)

(*******************************************************************)

PROCEDURE PLOTPOINTS;

     BEGIN
     (*$R APPLESTUFF *)                 (* Retain sound UNIT in memory  *)

          FOR INDEX:=1 TO N DO          (* Plot the points                  *)
               BEGIN
                    VALUE:=EIGVAL[INDEX];

                    IF (N>5) THEN
                         BEGIN
                              ROW:=ROUND(2*VALUE);
                              X:=11 + (INDEX * 4);
                         END
                    ELSE
                         BEGIN
                              ROW:=ROUND(4*VALUE);
                              X:=11 + (INDEX * 8);
                    END;

                Y:=21 - ROW;
                PITCH:=31 - Y;

                GOTOXY(X,Y);
                WRITE(CHR(15),'*',CHR(14));
                NOTE(PITCH,10);
            END;
    END;  (* End of PLOT POINTS *)

(*****************************************************************)

PROCEDURE SHOWINSTRUCTIONS;

    BEGIN
        WRITELN('When you are ready, you will be shown a plot of ',
                'Eigenvalue Magnitudes vs.');
        WRITELN('Factor Numbers.  You will be asked to ',
                'visualize a line passing through');
        WRITELN('the right most points and extending to the left.');
        WRITELN;
        WRITELN('The most significant factors are those that do ',
                 CHR(15),'not',CHR(14),' fall on that line ',
                 CHR(15),'plus',CHR(14));
        WRITELN('the first one that does.  That is the number of ',
                'factors you should keep.');
        GOTOXY(22,22);
        WRITE('Press any key to see the plot   ');
    END;  (* End of SHOW INSTRUCTIONS *)

(*****************************************************************)
(*                    Main body of SCREE routine                 *)
(*****************************************************************)

    BEGIN
    (*$I-*)
        WRITELN(CHR(12),' ':35,CHR(15),' SCREE TEST ',CHR(14));
        GOTOXY(0,5);

        SHOWINSTRUCTIONS;

        GETOPTION(OPT);
        ERASE(5,8);
        ERASE(22,1);

        DRAWAXIS;
        LABELAXIS;
        PLOTPOINTS;

        GOTOXY(37,8);
        WRITE('Visualize a ''scree line'' through');
        GOTOXY(37,9);
        WRITE('the right-most points.  Enter the');
        GOTOXY(37,10);
        WRITE('number of points not on that');
        GOTOXY(37,11);
        WRITE('line ',CHR(15),'plus',CHR(14),' 1: ',CHR(7));

        RESET(INPUT);
```

```
            READ(NS);

            WHILE (IORESULT=14) OR (NS<1) OR (NS>N) DO
                BEGIN
                    GOTOXY(37,14);
                    WRITE(' ',CHR(15),'WARNING:',CHR(14),' Must be ',
                            'at least 1 and');
                    GOTOXY(47,15);
                    WRITE('no more than ',N,' factors.');
                    GOTOXY(47,17);
                    WRITE('Press any key to try again');
                    GOTOXY(37,14);
                    GETOPTION(OPT);
                    GOTOXY(50,11);
                    WRITE(' ':20);
                    FOR I:=1 TO 4 DO
                        BEGIN
                            GOTOXY(37,(I+13));
                            WRITE(' ':36);
                        END;
                    GOTOXY(50,11);
                    RESET(INPUT);
                    READ(NS);
                END;  (* End of Bad number of factors *)

        (*$I+*)
        END;  (* End of SCREE test *)

(*****************************************************************************)


PROCEDURE SELECTFACTORS;

(*****************************************************************************)
(*                                                                           *)
(*          This procedure calculates and prints the percents of            *)
(*               variance explained by each factor, and then has            *)
(*               the user select the number of factors (NS) to              *)
(*               maintain.  Selection is done by one of these:              *)
(*                                                                           *)
(*               1 - Default (Eigenvalues > 1.0)                            *)
(*               2 - Scree Test                                             *)
(*               3 - Bartlett's Sphericity Test                             *)
(*               4 - User select                                            *)
(*                                                                           *)
(*****************************************************************************)

    VAR
        I,J,                            (* Iteration counters        *)
        N:                              (* Number of factors         *)
                INTEGER;
        VALUE,                          (* Eigenvalue                *)
        PCTOFVAR,                       (* Percent of variance       *)
        CUMPCT:                         (* Cumulative percent        *)
                REAL;
        OPT,                            (* Menu options              *)
        OPT1:
                CHAR;

(*****************************************************************************)
```

```
(*                       Internal Procedures                          *)
(********************************************************************)

   PROCEDURE GETNPRINTSTATS;

        BEGIN
             FOR I:=1 TO N DO
                  BEGIN
                       VALUE:=EIGVAL[I];
                       PCTOFVAR:=(VALUE/N)*100;
                       CUMPCT:=CUMPCT+PCTOFVAR;

                       WRITELN(I:4,VALUE:13:4,PCTOFVAR:12:1,
                                 CUMPCT:12:1);
                       IF (PRINTER) THEN
                            WRITELN(PTR,I:4,VALUE:13:4,PCTOFVAR:12:1,
                                      CUMPCT:12:1);
                  END;
             IF (PRINTER) THEN
                  WRITELN(PTR,CHR(13));
        END;   (* End of GET aNd PRINT STATisticS *)

(********************************************************************)

   PROCEDURE RECAPSELECTION;

        BEGIN
             WRITELN(CHR(12),' ':30,CHR(15),' FACTOR SELECTION ',
                       CHR(14));
             GOTOXY(0,15);
             CUMPCT:=0.0;
             FOR I:=1 TO NS DO
                  CUMPCT:=CUMPCT + (EIGVAL[I]/N) * 100;

             WRITE('You have selected ',NS,' factor');
             IF (NS>1) THEN
                  WRITE('s');
             WRITELN(' to continue analysis with.',CHR(13));
             WRITELN('This explains',CUMPCT:4:1,'% of the variance.');

             IF (PRINTER) THEN
                  BEGIN
                       WRITELN(PTR,NS,' factor(s) chosen to continue ',
                                 ' FACTOR analysis with.');
                       WRITELN(PTR,'This explains',CUMPCT:4:1,'% of ',
                                 'the variance.',CHR(13),CHR(13));
                       END;
        END;   (* End of RECAP SELECTION *)

(********************************************************************)
(*                    Main body of SELECT FACTORS                     *)
(********************************************************************)

   BEGIN
        N:=GROUP[0];                    (* Initialize parameters        *)
        NS:=0;
        CUMPCT:=0.0;

        GOTOXY(0,5);
        WRITELN(CHR(15),'FACTOR','EIGENVALUE':13,'PCT OF VAR':13,
            'CUM PCT':10,CHR(14),CHR(13));

      IF (PRINTER) THEN
         WRITELN(PTR,'FACTOR','EIGENVALUE':13,'PCT OF VAR':13,
                 'CUM PCT':10,CHR(13));

      GETNPRINTSTATS;

      GOTOXY(0,18);
      WRITELN('Pick desired method of FACTOR selection:');
      WRITELN('      1 - Eigenvalues > 1.0');
      WRITELN('      2 - Scree Test');
      WRITELN('      3 - Bartlett''s Sphericity Test');
      WRITELN('      4 - User selection');
      GETOPTION(OPT);
      WHILE (OPT<'1') OR (OPT>'4') DO
          GETOPTION(OPT);
      ERASE(18,5);

      CASE (OPT) OF                    (* Get significant factors *)
         '1':  FOR I:=1 TO N DO
                  IF (EIGVAL[I]>1.0) THEN
                     NS:=NS+1;
         '2':  SCREE(N,NS,EIGVAL);
         '3':  BARTLETT(N,NS,NUMREC,EIGVAL);
         '4':  USERSELECT(N,NS);
      END;  (* End of CASE *)

      RECAPSELECTION;

      GOTOXY(0,22);
      WRITE('Press any key to continue FACTOR analysis   ');
      GETOPTION(OPT1);
      GROUP[-1]:=NS;

      WRITELN(CHR(12),' ':26,CHR(15),' FACTOR ANALYSIS ROUTINE ',
              CHR(14));
   END;  (* End of SELECT FACTORS *)

(*******************************************************************)
(*                    Initialization part of UNIT                   *)
(*******************************************************************)

END.
```

```
(*$S+*)

UNIT MU_J; INTRINSIC CODE 20;

INTERFACE
    USES TRANSCEND, MAIN_UNIT, MU_E;

    PROCEDURE GETCVSS(VAR DATA:RAWDATA;VAR GROUP:HEADER2;
                      VAR ALPHA,BETA:MATRIX;NUMREC,WIDTH:INTEGER;
                      PRINTER:BOOLEAN);

    PROCEDURE STRUCTURECORR(VAR ALPHA,BETA,CM:MATRIX;VAR EIGVAL:VECTOR;
                            VAR SPECS1:HEADER1;VAR GROUP:HEADER2;
                            WIDTH:INTEGER;PRINTER:BOOLEAN);

IMPLEMENTATION

(*****************************************************************)
(*                       Main body of MU_J                        *)
(*****************************************************************)

PROCEDURE GETCVSS;

(*****************************************************************)
(*                                                                *)
(*          This procedure calculates the Canonical Variate       *)
(*               scores and then prints and/or saves them,        *)
(*               as desired.                                      *)
(*                                                                *)
(*               Y* = (Y) * Alpha        X* = (X) * Beta          *)
(*                                                                *)
(*****************************************************************)

    VAR
        I,J,L,                      (* Iteration counters         *)
        INDEX,                      (* Sorter & Eigen counter     *)
        P,                          (* Number of criterions       *)
        K,                          (* Number of predictors       *)
        N,                          (* Lessor of P and K          *)
        X,                          (* Index of X* in SCORES      *)
        Y:                          (* Index of Y* in SCORES      *)
            INTEGER;
        SPCS1:                      (* Field names of saved SCORES *)
            HEADER1;
        A,                          (* Sorted pointer array       *)
        SPCS2:                      (* Field widths of saved SCORES *)
            HEADER2;
        OPT1,                       (* Menu options               *)
        OPT2:
            CHAR;
        FIELD:                      (* Field number identifier    *)
            STRING;

(*****************************************************************)
(*                      Internal Procedures                       *)
(*****************************************************************)

    PROCEDURE CALCTHEVALUES;

        BEGIN
```

```
          FOR INDEX:=1 TO N DO        (* Calculate Can Var scores  *)
              FOR L:=1 TO NUMREC DO     (* For each eigenvalue     *)
                  BEGIN
                      X:=2*INDEX;
                      Y:=X-1;

                      SCORES[L,Y]:=0.0;             (* Get Y*       *)
                      FOR J:=1 TO P DO
                          IF (ALPHA[J,INDEX]=99.9999) THEN
                              SCORES[L,Y]:=99.9999
                          ELSE
                              IF (SCORES[L,Y]<>99.9999) THEN
                                  SCORES[L,Y]:=SCORES[L,Y]+
                                      DATA[L,A[J]]*ALPHA[J,INDEX];

                      SCORES[L,X]:=0.0;             (* Get X*       *)
                      FOR J:=1 TO K DO
                          IF (BETA[J,INDEX]=99.9999) THEN
                              SCORES[L,X]:=99.9999
                          ELSE
                              IF (SCORES[L,X]<>99.9999) THEN
                                 SCORES[L,X]:=SCORES[L,X]+
                                    DATA[L,A[(P+J)]]*BETA[J,INDEX];
                  END;
      END;  (* End of CALCulate THE VALueS *)

(******************************************************************)

    PROCEDURE PRINTTHEVALUES;

      BEGIN
          GOTOXY(8,3);
          WRITELN(CHR(15),' CANONICAL VARIATE SCORES ',CHR(14));
          GOTOXY(2,5);

          FOR I:=1 TO N DO                  (* Display headers        *)
              WRITE('CANVAR':13,I:2);
          GOTOXY(5,6);
          FOR I:=1 TO N DO
              WRITE('First  Second':15);
          WRITELN(CHR(13));

          IF (PRINTER) THEN                 (* Printer headers        *)
              BEGIN
                  WRITELN(PTR,'CANONICAL VARIATE SCORES:');
                  WRITE(PTR,CHR(13),'    ');

                  FOR I:=1 TO N DO
                      WRITE(PTR,'CANVAR':13,I:2);
                  WRITE(PTR,CHR(13),' ':5);
                  FOR I:=1 TO N DO
                      WRITE(PTR,'First  Second':15);
                  WRITELN(PTR,CHR(13));
              END;

          FOR L:=1 TO NUMREC DO             (* Display values         *)
              BEGIN
                  IF ((L > 1) AND (L MOD 13 = 1)) THEN
                      BEGIN                 (* Pause at page end      *)
                          GOTOXY(22,22);
```

```
                    WRITE('Press any key to continue   ');
                    GETOPTION(OPT2);
                    ERASE(8,15);
                    GOTOXY(0,8);
               END;  (* End of Pause *)

          WRITE(L:3,'  ');
          FOR INDEX:=1 TO N DO
               BEGIN
                    X:=2*INDEX;
                    Y:=X-1;
                    WRITE(SCORES[L,Y]:8:3,SCORES[L,X]:7:3);
               END;
          WRITELN;

          IF (PRINTER) THEN      (* Print values           *)
               BEGIN
                    WRITE(PTR,L:3,'  ');
                    FOR INDEX:=1 TO N DO
                         BEGIN
                              X:=2*INDEX;
                              Y:=X-1;
                              WRITE(PTR,SCORES[L,Y]:8:3,
                                        SCORES[L,X]:7:3);
                         END;
                    WRITELN(PTR);
               END;
     END;  (* End of Display values *)

     IF (PRINTER) THEN
          WRITELN(PTR,CHR(13));

     GOTOXY(22,22);
     WRITE('Done.  Press any key to continue   ');
     GETOPTION(OPT2);
     ERASE(3,21);
END;  (* End of PRINT THE VALUES *)

(*****************************************************************)

PROCEDURE SAVETHEVALUES;

     BEGIN
          GOTOXY(0,5);
          WRITELN('Make sure a preformatted Data disk is in ',
                  'Drive #2.  Press any key to continue');
          GETOPTION(OPT2);

          FOR INDEX:=1 TO N DO              (* Set field names    *)
               BEGIN
                    X:=2*INDEX;
                    Y:=X-1;
                    FIELD:=COPY('123456789',INDEX,1);
                    SPCS1[Y]:=CONCAT('  YCV',FIELD,'  ');
                    SPCS1[X]:=CONCAT('  XCV',FIELD,'  ');
               END;

          SPCS2[-1]:=NUMREC;                (* Set file specs     *)
          SPCS2[0]:=2*N;
          FOR INDEX:=1 TO (2*N) DO          (* Set field widths    *)
               SPCS2[INDEX]:=8;

          SAVEFILE(SCORES,SPCS1,SPCS2);

     END;  (* End of SAVE THE VALUES *)

(**************************************************************************)
(*                       Main body of CVSS routine                        *)
(**************************************************************************)

  BEGIN
     P:=GROUP[-1];                          (* Initialize parameters     *)
     K:=GROUP[0];

     IF (P>K) THEN
          N:=K
     ELSE
          N:=P;

     INDEX:=0;                              (* Set sequential pointers   *)
     FOR I:=1 TO WIDTH DO
          IF (GROUP[I]=1) THEN                   (* Criterion variable   *)
               BEGIN
                    INDEX:=INDEX+1;
                    A[INDEX]:=I;
               END;

     FOR I:=1 TO WIDTH DO
          IF (GROUP[I]=2) THEN                   (* Predictor variable   *)
               BEGIN
                    INDEX:=INDEX+1;
                    A[INDEX]:=I;
               END;

     CALCTHEVALUES;                              (* Calculate CV Scores  *)
     ERASE(22,1);

     GOTOXY(0,18);
     WRITELN('Select desired option:');
     WRITELN('         1 - Print ',NUMREC,' Canonical Variate ',
               'Scores');
     WRITELN('         2 - Save the scores to disk');
     WRITELN('         3 - Do both Print and Save');
     WRITELN('         4 - Do nothing.  Proceed to Canonical ',
               'Loadings');
     GETOPTION(OPT1);
     WHILE (OPT1<'1') OR (OPT1>'4') DO
          GETOPTION(OPT1);

     ERASE(18,5);
     IF (OPT1='1') OR (OPT1='3') THEN   (* Print all the values *)
          PRINTTHEVALUES;

     IF (OPT1='2') OR (OPT1='3') THEN   (* Save to disk          *)
          SAVETHEVALUES;

  END;  (* End of GET Canonical Variate ScoreS *)

(**************************************************************************)
```

```
PROCEDURE STRUCTURECORR;

(*****************************************************************)
(*                                                               *)
(*       This procedure calculates and prints the Structure      *)
(*            Correlations and the Indexes of Redundancy         *)
(*            based on the Canonical Variate Coefficients        *)
(*            (Alpha & Beta), the Eigenvalues, and the           *)
(*            Sample Correlation Matrix (CM).                    *)
(*                                                               *)
(*****************************************************************)

    VAR
        I,J,L,                        (* Iteration counters          *)
        INDEX,                        (* Index into arrays           *)
        P,                            (* Number of criterions        *)
        K,                            (* Number of predictors        *)
        N,                            (* Lesser of P and K           *)
        ROW:                          (* Row on screen               *)
            INTEGER;
        RYSQ,                         (* Total variance in Y from X  *)
        RXSQ:                         (* Total variance in X from Y  *)
            REAL;
        OPT:                          (* Menu option                 *)
            CHAR;
        NAME:                         (* Field or variable name      *)
            STRING;
        NAMES:                        (* Sorted names by type        *)
            HEADER1;
        VY,                           (* Individual variances in Y   *)
        VX:                           (* Individual variances in X   *)
            VECTOR;
        RYY,                          (* Criterion self-correlation  *)
        RXX,                          (* Predictor self-correlation  *)
        RY,                           (* YCANVARs                    *)
        RX:                           (* XCANVARs                    *)
            MATRIX;

(*****************************************************************)
(*                      Internal Procedures                      *)
(*****************************************************************)

    PROCEDURE SORTNAMES;

        BEGIN
            INDEX:=0;
            FOR I:=1 TO WIDTH DO          (* Sort names by type      *)
                IF (GROUP[I]=1) THEN            (* Criterions      *)
                    BEGIN
                        INDEX:=INDEX+1;
                        NAMES[INDEX]:=SPECS1[I];
                    END;

            FOR I:=1 TO WIDTH DO
                IF (GROUP[I]=2) THEN            (* Predictors      *)
                    BEGIN
                        INDEX:=INDEX+1;
                        NAMES[INDEX]:=SPECS1[I];
                    END;
        END;   (* End of SORT NAMES by type *)

(*******************************************************************)

    PROCEDURE GETYCVS;

        BEGIN
            GOTOXY(7,5);
            FOR I:=1 TO P DO              (* Calculate YCANVAR's      *)
                BEGIN
                    NAME:=NAMES[I];            (* Field headers       *)
                    IF (LENGTH(NAME)>8) THEN
                        NAME:=COPY(NAME,1,8);
                    WRITE(NAME:9);
                    IF (PRINTER) THEN
                        WRITE(PTR,NAME:9);
                END;

            ROW:=7;
            IF (PRINTER) THEN
                WRITELN(PTR,CHR(13));

            FOR I:=1 TO P DO                   (* RY = R(YY) * Alpha *)
                FOR J:=1 TO N DO
                    BEGIN
                        RY[I,J]:=0.0;
                        FOR L:=1 TO P DO
                            RY[I,J]:=RY[I,J]+RYY[I,L]*ALPHA[L,J];
                    END;

            FOR I:=1 TO P DO                   (* Print the RY's      *)
                BEGIN
                    GOTOXY(0,ROW);
                    WRITE('YCV':4,I,'  ');
                    IF (PRINTER) THEN
                        WRITE(PTR,'YCV':4,I,'  ');

                    FOR J:=1 TO N DO
                        BEGIN
                            WRITE(RY[I,J]:9:4);
                            IF (PRINTER) THEN
                                WRITE(PTR,RY[I,J]:9:4);
                        END;

                    ROW:=ROW+1;
                    IF (PRINTER) THEN
                        WRITELN(PTR);
                END;  (* End of Print the RY's *)

            IF (PRINTER) THEN
                WRITELN(PTR,CHR(13));
        END;  (* End of GET Y Canonical VariateS *)

(*******************************************************************)

    PROCEDURE GETXCVS;

        BEGIN
            FOR I:=1 TO K DO              (* Calculate the XCANVAR's *)
                BEGIN
```

```
                    NAME:=NAMES[(P+I)];
                    IF (LENGTH(NAME)>8) THEN
                        NAME:=COPY(NAME,1,8);
                    WRITE(NAME:9);
                    IF (PRINTER) THEN
                        WRITE(PTR,NAME:9);
                END;

            ROW:=ROW+2;
            IF (PRINTER) THEN
                WRITELN(PTR,CHR(13));

            FOR I:=1 TO K DO                    (* RX = R(XX) * Beta  *)
                FOR J:=1 TO N DO
                    BEGIN
                        RX[I,J]:=0.0;
                        FOR L:=1 TO K DO
                            RX[I,J]:=RX[I,J]+RXX[I,L]*BETA[L,J];
                    END;

            FOR I:=1 TO K DO                    (* Print the RX's     *)
                BEGIN
                    GOTOXY(0,ROW);
                    WRITE('XCV':4,I,'  ');
                    IF (PRINTER) THEN
                        WRITE(PTR,'XCV':4,I,'  ');

                    FOR J:=1 TO N DO
                        BEGIN
                            WRITE(RX[I,J]:9:4);
                            IF (PRINTER) THEN
                                WRITE(PTR,RX[I,J]:9:4);
                        END;

                    ROW:=ROW+1;
                    IF (PRINTER) THEN
                        WRITELN(PTR);
                END;  (* End of Print the RX's *)

            IF (PRINTER) THEN
                WRITELN(PTR,CHR(13));
        END;  (* End of GET X Canonical VariateS *)

(**************************************************************************)

    PROCEDURE CALCINDEXES;

        BEGIN
            RYSQ:=0.0;
            FOR INDEX:=1 TO N DO          (* Calculate Y variances *)
                BEGIN
                    VY[INDEX]:=0.0;
                    FOR I:=1 TO P DO
                        VY[INDEX]:=VY[INDEX]+SQR(RY[I,INDEX]);
                    VY[INDEX]:=(VY[INDEX]/P)*EIGVAL[INDEX];
                    RYSQ:=RYSQ+VY[INDEX];
                END;

            RXSQ:=0.0;
            FOR INDEX:=1 TO N DO          (* Calculate X variances *)
```

```
                BEGIN
                    VX[INDEX]:=0.0;
                    FOR I:=1 TO K DO
                        VX[INDEX]:=VX[INDEX]+SQR(RX[I,INDEX]);
                    VX[INDEX]:=(VX[INDEX]/K)*EIGVAL[INDEX];
                    RXSQ:=RXSQ+VX[INDEX];
                END;

        END;  (* End of CALCulate INDEXES of redundancy *)

(****************************************************************)

    PROCEDURE PRINTRESULTS;

        BEGIN
            GOTOXY(24,2);
            WRITELN(CHR(15),' INDEXES OF REDUNDANCY ',
                    CHR(14),CHR(13),CHR(13));

            IF (PRINTER) THEN
                WRITELN(PTR,'INDEXES OF REDUNDANCY:',CHR(13));

            FOR I:=1 TO N DO
                BEGIN
                    WRITELN('VY':10,I,'  =  ',VY[I]:6:4);
                    IF (PRINTER) THEN
                        WRITELN(PTR,'VY':10,I,'  =  ',VY[I]:6:4);
                END;

            WRITELN(' ':16,'-------',CHR(13),RYSQ:23:4,' of ',
                    'total variance',CHR(13));
            IF (PRINTER) THEN
                WRITELN(PTR,' ':16,'-------',CHR(13),RYSQ:23:4,
                        ' of total variance',CHR(13));

            FOR I:=1 TO N DO
                BEGIN
                    WRITELN('VX':10,I,'  =  ',VX[I]:6:4);
                    IF (PRINTER) THEN
                        WRITELN(PTR,'VX':10,I,'  =  ',VX[I]:6:4);
                END;

            WRITELN(' ':16,'-------',CHR(13),RXSQ:23:4,' of ',
                    'total variance');
            IF (PRINTER) THEN
                WRITELN(PTR,' ':16,'-------',CHR(13),RXSQ:23:4,
                        ' of total variance');

        END;  (* End of PRINT the RESULTS *)

(****************************************************************)
(*                  Main body of STRUCTURECORR                  *)
(****************************************************************)

    BEGIN
        P:=GROUP[-1];                   (* Initialize parameters      *)
        K:=GROUP[0];
        IF (P>K) THEN
            N:=K
        ELSE
```

```
            N:=P;

        FOR I:=1 TO P DO               (* Access self-correlations      *)
            FOR J:=1 TO P DO
                RYY[I,J]:=CM[I,J];

        FOR I:=1 TO K DO
            FOR J:=1 TO K DO
                RXX[I,J]:=CM[(P+I),(P+J)];

        SORTNAMES;                     (* Sort names by type            *)

        GOTOXY(5,3);
        WRITELN(CHR(15),' STRUCTURE CORRELATIONS ',CHR(14));

        IF (PRINTER) THEN              (* Printer heading               *)
            BEGIN
                WRITELN(PTR,'STRUCTURE CORRELATIONS:',CHR(13));
                WRITE(PTR,' ':7);
            END;

        GETYCVS;                       (* Calculate the YCANVAR's       *)

        ROW:=ROW+2;
        IF (PRINTER) THEN
            WRITE(PTR,' ':7);
        GOTOXY(7,ROW);

        GETXCVS;                       (* Calculate the XCANVAR's       *)

        ERASE(22,1);
        GOTOXY(16,22);
        WRITE('Press any key to get Indexes of Redundancy    ');
        GETOPTION(OPT);
        ERASE(3,20);
        GOTOXY(0,22);
        WRITE('Calculating Indexes of Redundancy. . .',
                'Please stand by    ');

        CALCINDEXES;                   (* Indexes of Redundancy         *)

        ERASE(22,1);
        PRINTRESULTS;                  (* Print Indexes of Redundancy *)

        GOTOXY(16,22);
        WRITE('Done.  Press any key to exit CANCOR    ');
        GETOPTION(OPT);
    END;  (* End of STRUCTURE CORRelations *)

(*******************************************************************)
(*                    Initialization part of UNIT                  *)
(*******************************************************************)

END.
```

```
(*$S+*)

UNIT MU_K; INTRINSIC CODE 21;

INTERFACE
    USES TRANSCEND, MAIN_UNIT, MU_E;

    PROCEDURE GETFACTSCORES(VAR DATA:RAWDATA;VAR COEF:MATRIX;
                            VAR GROUP:HEADER2;NUMREC,WIDTH:INTEGER;
                            PRINTER:BOOLEAN);

    PROCEDURE FACTORMAT(VAR EIGVAL:VECTOR;VAR EIGVEC,FACTCOEF:MATRIX;
                        VAR SPECS1:HEADER1;VAR GROUP:HEADER2;
                        WIDTH:INTEGER;PRINTER:BOOLEAN);

    PROCEDURE GETCANCORSTATS(VAR EIGVAL,CANCORS,WILKSL,CHISQR:VECTOR;
                             NUMREC,P,K:INTEGER;PRINTER:BOOLEAN);

IMPLEMENTATION

(*******************************************************************)
(*                        Main body of MU_K                        *)
(*******************************************************************)

PROCEDURE GETFACTSCORES;

(*******************************************************************)
(*                                                                 *)
(*      This procedure calculates the Factor Scores and then       *)
(*           prints and/or saves them, as desired.                 *)
(*                                                                 *)
(*                    F(i) = X * Alpha(i)                          *)
(*                                                                 *)
(*******************************************************************)

    VAR
        I,J,L,                  (* Iteration counters              *)
        N,                      (* Number of manifestations        *)
        NS:                     (* Number of Significant Factors *)
               INTEGER;
        SPCS1:                  (* Saved field names               *)
               HEADER1;
        A,                      (* Pointers to manifestations      *)
        SPCS2:                  (* Saved filed widths              *)
               HEADER2;
        JOB,                    (* Menu options                    *)
        OPT:
               CHAR;
        FIELD:                  (* Field number identifier         *)
               STRING;

(*******************************************************************)
(*                      Internal Procedures                        *)
(*******************************************************************)

    PROCEDURE CALCTHESCORES;

        BEGIN
            FOR L:=1 TO NUMREC DO
                FOR I:=1 TO NS DO
                BEGIN
                     SCORES[L,I]:=0.0;
                     FOR J:=1 TO N DO
                          SCORES[L,I]:=SCORES[L,I]+
                                          DATA[L,A[J]]*COEF[J,I];
                END;
        END;  (* End of CALCulate THE SCORES *)

(*****************************************************************)

    PROCEDURE PRINTTHESCORES;

        BEGIN
             ERASE(19,4);
             GOTOXY(30,3);
             WRITELN(CHR(15),' FACTOR SCORES ',CHR(14));

             GOTOXY(0,5);                (* Display the headers     *)
             WRITE(' ':10,'CASE ');
             FOR I:=1 TO NS DO
                  WRITE(' FACT ',I,' ');
             WRITELN(CHR(13));

             IF (PRINTER) THEN           (* Print the headers       *)
                  BEGIN
                       WRITELN(PTR,'FACTOR SCORES:',CHR(13));
                       WRITE(PTR,' ':10,'CASE ');
                       FOR I:=1 TO NS DO
                            WRITE(PTR,' FACT ',I,' ');
                       WRITELN(PTR,CHR(13));
                  END;

             FOR L:=1 TO NUMREC DO       (* Print all the scores    *)
                  BEGIN
                       IF ((L > 1) AND (L MOD 14 = 1)) THEN  (* Pause *)
                            BEGIN
                                 GOTOXY(22,22);
                                 WRITE('Press any key to continue   ');
                                 GETOPTION(OPT);
                                 ERASE(7,16);
                                 GOTOXY(0,7);
                            END;  (* End of Pause at page end *)

                       WRITE(L:13,'  ');
                       FOR I:=1 TO NS DO
                            WRITE(SCORES[L,I]:7:4,' ');
                       WRITELN;

                       IF (PRINTER) THEN
                            BEGIN
                                 WRITE(PTR,L:13,'  ');
                                 FOR I:=1 TO NS DO
                                      WRITE(PTR,SCORES[L,I]:7:4,' ');
                                 WRITELN(PTR);
                            END;
                  END;  (* End of Print all the scores *)

             GOTOXY(22,22);
             WRITE('Done.  Press any key to continue   ');
             GETOPTION(OPT);
```

```
          ERASE(3,20);

     END;  (* End of PRINT THE SCORES *)

(*******************************************************************)

  PROCEDURE SAVETHESCORES;

     BEGIN
          ERASE(19,4);
          GOTOXY(0,5);
          WRITE('Make sure a preformatted Data disk is ',
                'on-line.  Press any key to continue   ');
          GETOPTION(OPT);

          FOR I:=1 TO NS DO                (* Set field names     *)
               BEGIN
                    FIELD:=COPY('123456789',I,1);
                    SPCS1[I]:=CONCAT(' FACT ',FIELD,' ');
               END;

          SPCS2[-1]:=NUMREC;
          SPCS2[0]:=NS;

          FOR I:=1 TO NS DO
               SPCS2[I]:=8;

          SAVEFILE(SCORES,SPCS1,SPCS2);

     END;  (* End of SAVE THE SCORES *)

(*******************************************************************)
(*                   Main body of GETFACTSCORES                    *)
(*******************************************************************)

  BEGIN
     NS:=GROUP[-1];                    (* Initialize parameters    *)
     N:=GROUP[0];
     J:=0;

     FOR I:=1 TO WIDTH DO              (* Set sequential pointers *)
          IF (GROUP[I]>0) THEN
               BEGIN
                    J:=J+1;
                    A[J]:=I;
               END;

     CALCTHESCORES;                    (* Calculate Factor Scores *)
     ERASE(22,1);

     GOTOXY(0,18);
     WRITELN('Select desired option:');
     WRITELN('        1 - Print ',NUMREC,' Factor Scores');
     WRITELN('        2 - Save the scores to disk');
     WRITELN('        3 - Do both Print and Save');
     WRITELN('        4 - Do nothing. Exit FACTOR routine');
     GETOPTION(JOB);
     WHILE (JOB<'1') OR (JOB>'4') DO
          GETOPTION(JOB);
```

```
        ERASE(18,5);
        IF (JOB='1') OR (JOB='3') THEN
            PRINTTHESCORES;

        IF (JOB='2') OR (JOB='3') THEN
            SAVETHESCORES;

        GOTOXY(16,22);
        WRITE('Press any key to exit FACTOR routine    ');
        GETOPTION(OPT);

    END;  (* End of GET FACTor SCORES *)

(*******************************************************************)

PROCEDURE FACTORMAT;

(*******************************************************************)
(*                                                                 *)
(*        This procedure calculates and prints the Factor          *)
(*             Loadings, Communalities and Factor Score            *)
(*             Coefficients for each of the designated             *)
(*             variables under FACTOR analysis.                    *)
(*                                                                 *)
(*******************************************************************)

    VAR
        I,J,                          (* Iteration counters        *)
        INDEX,                        (* Index of designated field *)
        N,                            (* Number of total factors   *)
        NS:                           (* Number of selected factors *)
                  INTEGER;
        SUM:                          (* Sum of squares (normalize) *)
                  REAL;
        COMMUNAL:                     (* Total communalities       *)
                  VECTOR;
        FACTLOAD:                     (* Principal Factor Loadings  *)
                  MATRIX;
        OPT:                          (* Menu option               *)
                  CHAR;
        NAME,                         (* Name displayed on screen  *)
        NAMEP:                        (* Name printed on printer   *)
                  STRING;

(*******************************************************************)
(*                       Internal Procedures                       *)
(*******************************************************************)

    PROCEDURE NORMALIZE;                  (* Required eigenvectors *)

        BEGIN
            FOR J:=1 TO NS DO
                BEGIN
                    SUM:=0.0;
                    FOR I:=1 TO N DO
                        SUM:=SUM+SQR(EIGVEC[I,J]);
                    FOR I:=1 TO N DO
                        EIGVEC[I,J]:=EIGVEC[I,J]/SQRT(SUM);
                END;
        END;  (* End of NORMALIZE eigenvectors *)
```

```
(******************************************************************)

  PROCEDURE GETSTATS;

    BEGIN
      FOR I:=1 TO N DO
        FOR J:=1 TO NS DO
          FACTLOAD[I,J]:=SQRT(EIGVAL[J])*EIGVEC[I,J];

      FOR I:=1 TO N DO
        FOR J:=1 TO NS DO
          BEGIN
            COMMUNAL[I]:=COMMUNAL[I]+SQR(FACTLOAD[I,J]);
            FACTCOEF[I,J]:=FACTLOAD[I,J]/EIGVAL[J];

            IF (COMMUNAL[I] > 1.0) THEN
              COMMUNAL[I]:= 1.0;
          END;

    END;  (* End of GET STATisticS *)

(******************************************************************)

  PROCEDURE PRINTLOADINGS;

    BEGIN
      GOTOXY(9,7);                    (* Print headers          *)
      FOR I:=1 TO NS DO
        WRITE('  FACT ',I,' ');
      WRITELN(CHR(13));

      IF (PRINTER) THEN
        BEGIN
          WRITE(PTR,' ':17);
          FOR I:=1 TO NS DO
            WRITE(PTR,'     FACTOR ',I);
          WRITELN(PTR,CHR(13));
        END;

      INDEX:=0;                       (* Print the values       *)
      FOR I:=1 TO WIDTH DO
        IF (GROUP[I]>0) THEN          (* Designated variable    *)
          BEGIN
            INDEX:=INDEX+1;
            NAMEP:=SPECS1[I];

            IF (LENGTH(NAMEP)>15) THEN
              NAMEP:=COPY(NAMEP,1,15);

            IF (LENGTH(NAMEP)>8) THEN
              NAME:=COPY(NAMEP,1,8)
            ELSE
              NAME:=NAMEP;

            WRITE(NAME:8,' ');
            IF (PRINTER) THEN
              WRITE(PTR,NAMEP:15,'    ');

            FOR J:=1 TO NS DO
```

```
                                BEGIN
                                    WRITE(' ',FACTLOAD[INDEX,J]:6:4,' ');
                                    IF (PRINTER) THEN
                                        WRITE(PTR,'    ',
                                              FACTLOAD[INDEX,J]:8:4,'   ');
                                END;

                            WRITELN;
                            IF (PRINTER) THEN
                                WRITELN(PTR);
                        END;
            END;   (* End of PRINT factor LOADINGS *)

(*************************************************************************)

        PROCEDURE PRINTCOMMUNALITIES;

            BEGIN
                GOTOXY(0,5);
                WRITELN('VARIABLE':17,'COMMUNALITY':16);
                IF (PRINTER) THEN
                    WRITELN(PTR,CHR(13),'VARIABLE':17,
                           'COMMUNALITY':16,CHR(13));
                GOTOXY(0,7);

                INDEX:=0;                           (* Print the values    *)
                FOR I:=1 TO WIDTH DO
                    IF (GROUP[I]>0) THEN
                        BEGIN
                            INDEX:=INDEX+1;
                            NAME:=SPECS1[I];

                            IF (LENGTH(NAME)>15) THEN
                                NAME:=COPY(NAME,1,15);

                            WRITELN(NAME:15,' ':5,'    ',
                                    COMMUNAL[INDEX]:5:4,'    ');
                            IF (PRINTER) THEN
                                WRITELN(PTR,NAME:15,' ':5,'    ',
                                        COMMUNAL[INDEX]:5:4,'    ');
                        END;

                IF (PRINTER) THEN
                    WRITELN(PTR,CHR(13));
            END;   (* End of PRINT COMMUNALITIES *)

(*************************************************************************)

        PROCEDURE PRINTCOEFFICIENTS;

            BEGIN
                GOTOXY(0,5);
                WRITELN('FACTOR SCORE COEFFICIENTS:');
                IF (PRINTER) THEN
                    WRITELN(PTR,'FACTOR SCORE COEFFICIENTS:',CHR(13));

                GOTOXY(9,7);                        (* Print headers        *)
                FOR I:=1 TO NS DO
                    WRITE('  FACT ',I,' ');
                WRITELN(CHR(13));
```

```
          IF (PRINTER) THEN
               BEGIN
                    WRITE(PTR,' ':19);
                    FOR I:=1 TO NS DO
                         WRITE(PTR,'   FACTOR ',I,' ');
                    WRITELN(PTR,CHR(13));
               END;

          INDEX:=0;                          (* Print the values       *)
          FOR I:=1 TO WIDTH DO
               IF (GROUP[I]>0) THEN          (* Designated variable    *)
                    BEGIN
                         INDEX:=INDEX+1;
                         NAMEP:=SPECS1[I];

                         IF (LENGTH(NAMEP)>15) THEN
                              NAMEP:=COPY(NAMEP,1,15);

                         IF (LENGTH(NAMEP)>8) THEN
                              NAME:=COPY(NAMEP,1,8)
                         ELSE
                              NAME:=NAMEP;

                         WRITE(NAME:8,' ');
                         IF (PRINTER) THEN
                              WRITE(PTR,NAMEP:15,'      ');

                         FOR J:=1 TO NS DO
                              BEGIN
                                   WRITE(' ',FACTCOEF[INDEX,J]:6:4,' ');
                                   IF (PRINTER) THEN
                                        WRITE(PTR,'   ',
                                               FACTCOEF[INDEX,J]:6:4,'  ');
                              END;

                         WRITELN;
                         IF (PRINTER) THEN
                              WRITELN(PTR);
                    END;

          IF (PRINTER) THEN
               WRITELN(PTR,CHR(13));
     END;  (* End of PRINT factor score COEFFICIENTS *)

(***************************************************************)
(*                  Main body of FACTOR MATrix                  *)
(***************************************************************)

    BEGIN
    (*$R TRANSCEND *)                    (* Retain UNIT in memory    *)

         NS:=GROUP[-1];                  (* Initialize parameters    *)
         N:=GROUP[0];
         FOR I:=1 TO N DO
              COMMUNAL[I]:=0.0;

         NORMALIZE;

         GETSTATS;
```

```
          ERASE(22,1);

          GOTOXY(0,5);
          WRITELN('FACTOR MATRIX USING PRINCIPAL FACTOR(S):');
          IF (PRINTER) THEN
              WRITELN(PTR,'FACTOR MATRIX USING PRINCIPAL FACTOR(S):',
                      CHR(13));

          PRINTLOADINGS;                      (* Factor loadings         *)

          GOTOXY(0,22);
          WRITE(CHR(29),'Press any key to print Communalities   ');
          GETOPTION(OPT);
          ERASE(5,18);

          PRINTCOMMUNALITIES;                 (* Total explained         *)

          GOTOXY(0,22);
          WRITE('Press any key to print Factor Score Coefficients   ');
          GETOPTION(OPT);
          ERASE(5,18);

          PRINTCOEFFICIENTS;                  (* Factor coefficients     *)

          GOTOXY(22,22);
          WRITE('Press any key to continue   ');
          GETOPTION(OPT);
          ERASE(5,18);

          IF (PRINTER) THEN
              FOR I:=1 TO 3 DO
                  WRITELN(PTR);
     END;  (* End of FACTOR MATrix *)

(*****************************************************************)

PROCEDURE GETCANCORSTATS;

(*****************************************************************)
(*                                                               *)
(*        This procedure calculates the Canonical Correlation,   *)
(*              Wilk's Lambda, and Chi Square statistics from    *)
(*              the Eigenvalues and then prints them all.        *)
(*                                                               *)
(*****************************************************************)

     VAR
          I,                                  (* Iteration counter       *)
          INDEX,                              (* One of 'N' values       *)
          N:                                  (* Lessor of P and K       *)
                 INTEGER;
          MULT,                               (* Multiplication constant *)
          VALUE:                              (* Calculated Wilk's Lambda *)
                 REAL;
          OPT:                                (* Menu option             *)
                 CHAR;

(*****************************************************************)
(*                    Internal Procedures                        *)
(*****************************************************************)
```

```
    PROCEDURE CALCSTATS;

        BEGIN
            MULT:=-(NUMREC-1-(P+K+1)/2.0);
            FOR INDEX:=1 TO N DO
                BEGIN
                    IF (EIGVAL[INDEX]>0.0) THEN
                        CANCORS[INDEX]:=SQRT(EIGVAL[INDEX])
                    ELSE
                        CANCORS[INDEX]:=99.9999;

                    VALUE:=1.0;
                    FOR I:=INDEX TO N DO
                        VALUE:=VALUE*(1.0-EIGVAL[I]);

                    WILKSL[INDEX]:=VALUE;

                    IF (VALUE<=0.0) THEN
                        CHISQR[INDEX]:=99.9999
                    ELSE
                        CHISQR[INDEX]:=MULT*LN(VALUE);
                END;
        END;  (* End of CALCulate the STATisticS *)

(*******************************************************************)

    PROCEDURE PRINTHEADINGS;

        BEGIN
            GOTOXY(13,5);
            WRITELN(CHR(15),' CANONICAL CORRELATION ',CHR(14));
            GOTOXY(0,8);
            WRITELN('CANONICAL':30,'WILK''S':9,'CHI-':8);
            WRITELN('NUMBER','EIGENVALUE':12,'CORRELATION':13,
                    'LAMBDA':8,'SQUARE':9,CHR(13));

            IF (PRINTER) THEN                     (* Printer Headings *)
                BEGIN
                    WRITELN(PTR,'CANONICAL':30,'WILK''S':9,'CHI-':8);
                    WRITELN(PTR,'NUMBER','EIGENVALUE':12,
                            'CORRELATION':13,'LAMBDA':8,'SQUARE':9);
                    WRITELN(PTR);
                END;
        END;  (* End of PRINT the HEADINGS *)

(*******************************************************************)

    PROCEDURE PRINTSTATS;

        BEGIN
            FOR INDEX:=1 TO N DO              (* Output statistics    *)
                BEGIN
                    WRITELN(INDEX:3,EIGVAL[INDEX]:13:4,
                            CANCORS[INDEX]:12:4,WILKSL[INDEX]:11:4,
                            CHISQR[INDEX]:9:4);

                    IF (PRINTER) THEN
                        WRITELN(PTR,INDEX:3,EIGVAL[INDEX]:13:4,
                                CANCORS[INDEX]:12:4,
                                        WILKSL[INDEX]:11:4,
                                        CHISQR[INDEX]:9:4);
                            END;

                         IF (PRINTER) THEN
                            WRITELN(PTR,CHR(13));
                      END;   (* End of PRINT the STATisticS *)

(*****************************************************************)
(*                        Main body of GETSTATS                  *)
(*****************************************************************)

   BEGIN
   (*$R TRANSCEND *)                         (* Retain UNIT in memory     *)

      IF (P>K) THEN                          (* Initialize parameters     *)
         N:=K
      ELSE
         N:=P;

      CALCSTATS;                             (* Calculate the statistics *)

      GOTOXY(16,22);
      WRITE('Done.  Press any key to print results.   ');
      GETOPTION(OPT);
      ERASE(20,3);

      PRINTHEADINGS;

      PRINTSTATS;                            (* Print the statistics      *)

   END;  (* End of GET CANCOR STATisticS *)

(*****************************************************************)
(*                    Initialization part of UNIT                *)
(*****************************************************************)

END.
```

BIBLIOGRAPHY

1. Apple Pascal: Language Reference Manual (Apple II). Apple Product #A2L0028. Cupertino, CA: Apple Computer Inc., 1980.

2. Apple Pascal: Operating System Reference Manual (Apple II). Apple Product #A2L0027. Cupertino, CA: Apple Computer Inc., 1980.

3. Carnahan, Brice, H.A. Luther, and James O. Wilkes. Applied Numerical Methods. New York: John Wiley & Sons, Inc., 1969.

4. Coleman, Maj Joseph W. Class notes from SM6.85, Applied Multivariate Data Analysis. School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, OH, Winter Quarter, 1983.

5. Conte, S.D. and Carl de Boor. Elementary Numerical Analysis (Second edition). New York: McGraw-Hill Book Company, 1972.

6. Cox, Jeff G. "9 Lessons in Fault Tolerant Software", Nibble, 4 (3): 147-151 (March 1983).

7. Douglass, Bruce P. "Copernica Mathematica", 80 Micro, 43: 328-339 (August 1983).

8. Kamins, Scot and Mitchell Waite. Apple Backpack: Humanized Programming in Basic. Peterborough, NH: BYTE Books, 1982.

9. Korites, B.J. Graphic Software for Microcomputers. Duxbury, MA: Kern Publications, 1981.

10. Lewis, T.G. and M.Z. Smith. Applying Data Stuctures. Boston: Houghton Mifflin Company, 1976.

11. Luehrmann, Arthur and Herbert Peckham. Apple Pascal: A Hands-On Approach. New York: McGraw-Hill Book Company, 1981.

12. McMillan, Claude and Richard F. Gonzales. Systems Analysis: A Computer Approach to Decision Models (Third edition). Homewood, IL: Richard D. Irwin, Inc., 1973.

13. McNichols, Charles W. An Introduction to Applied Multivariate Data Analysis. Unpublished Course Notes, SM6.85. School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, OH, 1980.

14. Merritt, Jim. "The Pascal Path: Jungle Fever, Part 9," Softalk, 4: 205-214 (November 1983).

15. -----. "The Pascal Path: Jungle Fever, Part 8," Softalk, 4: 227-238 (October 1983).

16. -----. "The Pascal Path: Jungle Fever, Part 7," Softalk, 4: 241-250 (September 1983).

17. Nie, Norman H. et al. Statistical Package for the Social Sciences (Second edition). New York: McGraw-Hill Book Company, 1975.

18. Root, Jock. "A Custom Menu Generator," Softalk, 3: 165-169 (May 1983).

19. Swanson, Paul. Microcomputer Disk Techniques. Peterborough, NH: BYTE/McGraw-Hill, Inc., 1982.

20. Zaks, Rodnay. Introduction to PASCAL (Second edition revised). Berkley, CA: SYBEX Inc., 1981.

VITA

Captain David P. Kunkel was born on 14 July 1954 in Nampa, Idaho. He graduated from high school in Urbana, Illinois, in 1972 and attended the United States Air Force Academy from which he received the degree of Bachelor of Science in Computer Science in June 1976. After attending Initial Qualification Training at Vandenberg AFB, California, he served as a Minuteman Missile Combat Crewmember in the 319th Strategic Missile Squadron and as an Instructor Crew Commander in the 90th Strategic Missile Wing/Training Division at F.E. Warren AFB, Wyoming until May 1981. During that time he earned the degree of Master of Business Administration from the University of Wyoming. He was then assigned to the 13th Missile Warning Squadron at Clear AFS, Alaska where he served as a Space Systems Director and as Chief, Standardization and Evaluation Section until entering the School of Engineering, Air Force Institute of Technology, in June 1982.

Permanent address: 406 E. Pennsylvania Ave
Urbana, Illinois 61801

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) AFIT/GSO/OS/83D-4	5. MONITORING ORGANIZATION REPORT NUMBER(S)

6a. NAME OF PERFORMING ORGANIZATION School of Engineering	6b. OFFICE SYMBOL *(If applicable)* AFIT/ENS	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS *(City, State and ZIP Code)* Air Force Institute of Technology Wright-Patterson AFB, Ohio 45433		7b. ADDRESS *(City, State and ZIP Code)*
8a. NAME OF FUNDING/SPONSORING ORGANIZATION	8b. OFFICE SYMBOL *(If applicable)*	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER

8c. ADDRESS *(City, State and ZIP Code)*	10. SOURCE OF FUNDING NOS.			
	PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT NO.
11. TITLE *(Include Security Classification)* PASCAL STATISTICAL PROCEDURES PACKAGE				

12. PERSONAL AUTHOR(S)
David P. Kunkel, MBA, Captain, USAF

13a. TYPE OF REPORT MS Thesis	13b. TIME COVERED FROM ________ TO ________	14. DATE OF REPORT *(Yr., Mo., Day)* 1983 December 16	15. PAGE COUNT 224

16. SUPPLEMENTARY NOTATION

Approved for public release: IAW AFR 190-17.
[illegible]
LYNN E. WOLAVER
Dean for Research and Professional Development
Air Force Institute of Technology (AIC)
Wright-Patterson AFB OH 45433

17. COSATI CODES		
FIELD	GROUP	SUB. GR.
12	01	

18. SUBJECT TERMS *(Continue on reverse if necessary and identify by block number)*
Canonical Correlation Analysis, Factor Analysis, Microcomputer, Multivariate Statistics, PASCAL

19. ABSTRACT *(Continue on reverse if necessary and identify by block number)*

Thesis Advisor: Joseph W. Coleman, Major, USAF

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT. ☐ DTIC USERS ☐	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED	
22a. NAME OF RESPONSIBLE INDIVIDUAL Joseph W. Coleman, Major, USAF	22b. TELEPHONE NUMBER *(Include Area Code)* 513-255-2549	22c. OFFICE SYMBOL AFIT/ENS

DD FORM 1473, 83 APR EDITION OF 1 JAN 73 IS OBSOLETE

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

This study showed that a set of procedures could be written and combined into a multivariate data analysis package that will run on a microcomputer. This package can be used as a teaching aid in the classroom or micro-computer center and as a research tool for users to do a ball-park analysis of a data base. Included in the package are procedures to handle data base definition and modification, Factor analysis, and Canonical Correlation analysis.

The PASCAL Statistical Procedures Package (PSPP) was written on an Apple IIe microcomputer using the Apple PASCAL language and operating system. It will output to a printer in a 132 character per line format. If an on-line printer is only capable of 80 characters per line, wrap-around will occur.

The package is composed of 4 top-level procedures stored in Regular Units and 163 library procedures stored in 13 Intrinsic Units. Units are Apple PASCAL structures that allow for program segmentation.